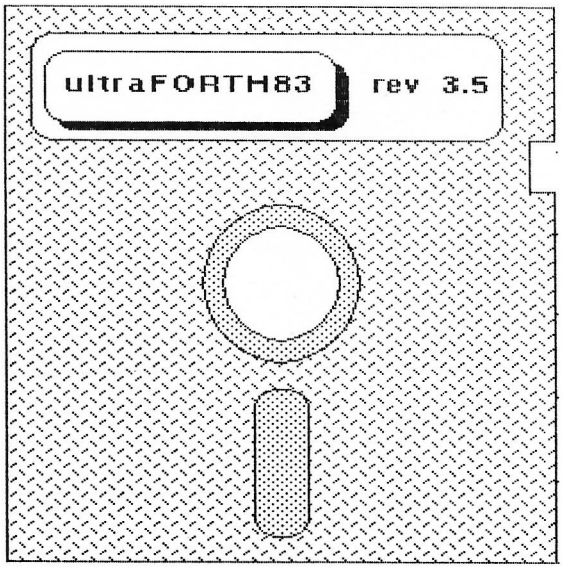
Glossare



Notation

Die Worte des ultraFORTH83 sind in Wortgruppen zusammengefasst. Die Worte jeder Sachgruppe sind alphabetisch sortiert. Die Wortgruppen sind nicht geordnet.

Worte werden in der in Kapitel 1 angegeben Schreibweise aufgefuehrt. Wortnamen im Text werden gross geschrieben.

Die Wirkung des Wortes auf den Stack wird in Klammern angegeben und zwar in folgender Form:

```
( vorher -- nachher )
```

vorher : Werte auf dem Stack vor Ausfuehrung des Wortes
nachher : Werte auf dem Stack nach Ausfuehrung des Wortes

In dieser Notation wird das oberste Element des Stacks immer ganz rechts geschrieben. Sofern nicht anders angegeben, beziehen sich alle Stacknotationen auf die spaetere Ausfuehrung des Wortes. Bei immediate Worten wird auch die Auswirkung des Wortes auf den Stack waehrend der Kompilierung angegeben.

Worte werden ferner durch folgende Symbole gekennzeichnet:

C
: Dieses Wort kann nur waehrend der Kompilation einer :-Definition benutzt werden.

I
: Dieses Wort ist ein immediate Wort, das auch im kompilierenden Zustand ausgefuehrt wird.

83
: Dieses Wort wird im 83-Standard definiert und muss auf allen Standardsystemen aequivalent funktionieren.

U
: Kennzeichnet eine Uservariable. Wird aufgrund eines Missverstaendnisses nicht konsequent verwendet.

Weicht die Aussprache eines Wortes von der natuerlichen englischen Aussprache ab, so wird sie in Anfuehrungszeichen angegeben. Gelegentlich folgt auch eine deutsche Uebersetzung.

Die Namen der Stackparameter folgen, sofern nicht suggestive Bezeichnungen gewaehlt wurden, dem nachstehendem Schema. Die Bezeichnungen koennen mit einer nachfolgenden Ziffer versehen sein.

Stack-Zahlentyp not.		Wertebereich in Dezimal	minimale Feldbreite
flag	logischer Wert	0=falsch, sonst=true	16 Bit
true	logischer Wert	-1 (als Ergebnis)	16
false	logischer Wert	0	16
b	Bit	0..1	1
char	Zeichen	0..127 (0..256)	7
8b	8 beliebige Bits	nicht anwendbar	8
16b	16 beliebige Bits	nicht anwendbar	16
n	Zahl, bewertete Bits	-32768..32767	16
+n	positive Zahl	0..32767	16
u	vorzeichenlose Zahl	0..65535	16
w	Zahl, n oder u	-32768..65535	16
addr	Adresse, wie u	0..65535	16
32b	32 beliebige Bits	nicht anwendbar	32
d	doppelt genaue Zahl	-2,147,483,648... 2,147,483,647	32
+d	pos. doppelte Zahl	0..2,147,483,647	32
ud	vorzeichenlose doppelt genaue Zahl	0..4,294,967,295	32
sys	0, 1 oder mehr System-abhaengige Werte	nicht anwendbar	na

Arithmetik

* (w1 w2 -- w3) 83
"mal", engl."times"
Der Werte w1 wird mit w2 multipliziert. w3 sind die niederwertigen 16 Bits des Produktes. Ein Ueberlauf wird nicht angezeigt.

*/ (n1 n2 n3 -- n4) 83
"mal-durch", engl."times-divide"
Zuerst wird n1 mit n2 multipliziert und ein 32-bit Zwischenergebnis erzeugt. n4 ist der Quotient aus dem 32-bit-Zwischenergebnis und dem Divisor n3. Das Produkt von n1 mal n2 wird als 32-bit Zwischenergebnis dargestellt, um eine groessere Genauigkeit gegenueber dem sonst gleichwertigen Ausdruck n1 n2 * n3 / zu erhalten. Eine Fehlerbedingung besteht, wenn der Divisor Null ist, oder der Quotient ausserhalb des Intervalls (-32768 .. 32767) liegt.

*/mod (n1 n2 n3 -- n4 n5) 83
"mal-durch-mod",engl."times-divide-mod"
Zuerst wird n1 mit n2 multipliziert und ein 32-bit Zwischenergebnis erzeugt. n4 ist der Rest und n5 der Quotient aus dem 32-bit-Zwischenergebnis und dem Divisor n3. n4 hat das gleiche Vorzeichen wie n3 oder ist Null. Das Produkt von n1 mal n2 wird als 32-bit Zwischenergebnis dargestellt, um eine groessere Genauigkeit gegenueber dem sonst gleichwertigen Ausdruck n1 n2 * n3 /mod zu erhalten. Eine Fehlerbedingung besteht, wenn der Divisor Null ist, oder der Quotient ausserhalb des Intervalls (-32768..32767) liegt.

\+ (w1 w2 -- w3) 83
"plus"
w1 und w2 addiert ergibt w3.

\- (w1 w2 -- w3) 83
"minus"
w2 von w1 subtrahiert ergibt w3 .

-1 (-- -1)
Oft benutzte Zahlenwerte wurden zu Konstanten gemacht. Definiert in der Form:
n CONSTANT n
Dadurch wird Speicherplatz eingespart und die Ausfuehrungszeit verkuerzt. Siehe auch CONSTANT .

/ (n1 n2 -- n3) 83
"durch",engl."divide"
n3 ist der Quotient aus der Division von n1 durch den Divisor n2. Eine Fehlerbedingung besteht, wenn der Divisor Null ist oder der Quotient ausserhalb des Intervalls (-32768 .. 32767) liegt.

/mod (n1 n2 -- n3 n4) 83
"durch-mod", engl."divide-mod"
n3 ist der Rest und n4 der ganzzahlige Quotient aus der Division von n1 durch den Divisor n2. n3 hat das selbe Vorzeichen wie n2 oder ist Null. Eine Fehlerbedingung besteht, wenn der Divisor Null ist oder der Quotient ausserhalb des Intervalls (-32768 .. 32767) liegt.

0 (-- 0)
Siehe -1 .

1 (-- 1)
Siehe -1 .

1+ (w1 -- w2) 83
"eins-plus", engl."one-plus"
w2 ist das Ergebnis von Eins plus w1. Die Operation 1 + wirkt genauso.

1- (w1 -- w2) 83
"eins-minus", engl."one-minus"
w2 ist das Ergebnis von Eins minus w1. Die Operation 1 - wirkt genauso.

2 (-- 2)
Siehe -1 .

2* (w1 -- w2)
"zwei-mal", eng.l"two-times"
w1 wird um ein Bit nach Links verschoben und das ergibt w2. In das niederwertigste Bit wird eine Null geschrieben. Die Operation 2 * wirkt genauso.

2+ (w1 -- w2) 83
"zwei-plus", engl."two-plus"
w2 ist das Ergebnis von Zwei plus w1. Die Operation 2 + wirkt genauso.

2- (w1 -- w2) 83
"zwei-minus", engl."two-minus"
w2 ist das Ergebnis von Zwei minus w1. Die Operation 2 - wirkt genauso.

2/ (n1 -- n2) 83
"zwei-durch", engl."two-divide"
n1 wird um ein Bit nach rechts verschoben und das ergibt n2 . Das Vorzeichen wird beruecksichtigt und bleibt unveraendert. Die Operation 2 / wirkt genauso.

3 (-- 3)
Siehe -1 .

3+ (w1 -- w2)
"drei-plus", engl."three-plus"
w2 ist das Ergebnis von Drei plus w1. Die Operation 3 + wirkt genauso.

4 (-- 4)
Siehe -1 .

abs (n -- u) 83
"absolut", engl."absolute"
u ist der Betrag von n . Wenn n gleich -32768 ist, hat u den selben Wert wie n. Vergleiche auch "Arithmetik, Zweierkomplement".

max (n1 n2 -- n3) 83
"maximum"
n3 ist der Groessere der beiden Werte n1 und n2. Benutzt die > Operation. Die groesste Zahl fuer n1 oder n2 ist 32767.

min (n1 n2 -- n3) 83
"minimum"
n3 ist der Kleinere der beiden Werte n1 und n2. Benutzt die < Operation. Die kleinste Zahl fuer n1 oder n2 ist -32768.

mod (n1 n2 -- n3) 83
"mod"
n3 ist der Rest der Division von n1 durch den Divisor n2. n3 hat dasselbe Vorzeichen wie n2 oder ist Null. Eine Fehlerbedingung besteht wenn der Divisor Null ist oder der Quotient ausserhalb des Intervals (-32768 .. 32767) liegt.

negate (n1 -- n2) 83
n2 hat den gleichen Betrag, aber das umgekehrte Vorzeichen von n1. n2 ist gleich der Differenz von Null minus n1.

u/mod (u1 u2 -- u3 u4)
"u-durch-mod", engl."u-divide-mod"
u3 ist der Rest und u4 der Quotient aus der Division von u1 durch den Divisor u2. Die Zahlen u sind vorzeichenlose 16-Bit-Zahlen (unsigned integer). Eine Fehlerbedingung besteht, wenn der Divisor Null ist.

umax (u1 u2 -- u3)
"u-maximum"
u3 ist der Groessere der beiden Werte u1 und u2. Benutzt die u> Operation. Die groesste Zahl fuer n1 oder n2 ist 65535.

umin (u1 u2 -- u3)
"u-minimum"
u3 ist der Kleinere der beiden Werte u1 und u2. Benutzt die u< Operation. Die kleinste Zahl fuer n1 oder n2 ist 0.

Logik und Vergleiche

```
0<              ( n -- flag )       83
"null-kleiner", engl."zero-less"
Wenn n kleiner als Null (negativ) ist, ist flag wahr. Dies ist
immer dann der Fall, wenn das hoechstwertigste Bit gesetzt ist.
Deswegen kann dieser Operator zu Test dieses Bits benutzt werden.

0<>             ( n -- flag )
Wenn n verschieden von Null ist, ist flag wahr.

0=              ( w -- flag )       83
"null-gleich", engl."zero-equals"
Wenn w gleich Null ist, ist flag wahr.

0>              ( n -- flag )       83
"null-groesser", engl."zero-greater"
Wenn n groesser als Null ist, ist flag wahr.

<               ( n1 n2 -- flag )   83
"kleiner-als", engl."less-than"
Wenn n1 kleiner als n2 ist, ist flag wahr.
Z.B.  -32768 32767 <   ist wahr.
      -32768     0 <   ist wahr.

=               ( w1 w2 -- flag )   83
"gleich", engl."equals"
Wenn w1 gleich w2 ist, ist flag wahr.

>               ( n1 n2 -- flag )   83
"groesser-als", engl."greater-than"
Wenn n1 groesser als n2 ist, ist flag wahr.
 Z.B.  -32768 32767 >    ist falsch.
       -32768     0 >    ist falsch.

and             ( n1 n2 -- n3 )     83
n1  wird mit n2 bitweise logisch UND verknuepft und das ergibt n3.

case?           ( 16b1 16b2 -- 16b1 false )
     oder       ( 16b1 16b2 -- true       )
"case-frage", engl."case-question"
Vergleicht die beiden obersten Werte auf dem Stack. Sind sie
gleich, verbleibt TRUE auf dem Stack. Sind sie verschieden,
verbleibt FALSE und der darunterliegende Wert 16b1 auf dem Stack.
Wird zB. benutzt in der Form:
KEY ASCII A CASE? IF ... EXIT THEN
    ASCII B CASE? IF ... EXIT THEN
    ..
    DROP
Entspricht dem Ausdruck  OVER = DUP IF NIP THEN.

false           ( -- 0 )
Hinterlaesst Null als Zeichen fuer logisch-falsch auf dem Stack.

not             ( n1 -- n2 )        83
Jedes Bit von n1 wird einzeln invertiert und das ergibt n2.

or              ( n1 n2 -- n3 )     83
n1 wird mit n2 bitweise logisch ODER verknuepft und das ergibt n3.
```

true (-- -1)
Hinterlaesst -1 als Zeichen fuer logisch-wahr auf dem Stack.

u< (u1 u2 -- flag) 83
"u-kleiner-als", engl."u-less-than"
Wenn u1 kleiner als u2 ist, ist flag wahr. Die Zahlen u sind vorzeichenlose 16-Bit-Zahlen. Wenn Adressen verglichen werden sollen, muss U< benutzt werden. Sonst passieren oberhalb von 32K eigenartige Dinge!

u> (u1 u2 -- flag)
"u-groesser-als", engl."u-greater-than"
Wenn u1 groesser als u2 ist, ist flag wahr, sonst gilt das gleiche wie fuer U< .

uwithin (n u1 u2 -- flag)
"u-within"
Wenn u1 kleiner oder gleich n ist und n kleiner u2 ist (u1<=n<u2), ist flag wahr. Benutzt die U< Operation.

xor (n1 n2 -- n3) 83
"x-or"
n1 wird mit n2 bitweise logisch EXCLUSIV ODER verknuepft und ergibt n3.

Speicheroperationen

! (16b adr --) 83
"store"
16b werden in den Speicher auf die Adresse adr geschrieben.
In 8-bitweise adressierten Speichern werden die zwei Bytes adr und adr+1 mit dem Wert 16b ueberschrieben. Der 6502 verlangt, das vom 16b Wert die niederwertigen 8b nach adr und die hoeerwertigen 8b nach adr+1 geschrieben werden (lowbyte-highbyte-order).

+! (w1 adr --) 83
"plus-store"
w1 wird zu dem Wert w in der Adresse adr addiert. Benutzt die + Operation. Die Summe wird in den Speicher in die Adresse adr geschrieben. Der alte Speicherinhalt wird ueberschrieben.

@ (adr -- 16b) 83
"fetch"
Von der Adresse adr wird der Wert 16b aus dem Speicher geholt. Dabei werden beim 6502 die niederwertigen 8b von adr und die hoeerwertigen 8b von adr+1 geholt und als 16b auf den Stack gelegt.

c! (16b adr --) 83
"c-store"
Von 16b werden die niederwertigen 8b in den Speicher in die Adresse adr geschrieben.

c@ (adr -- 8b) 83
"c-fetch"
Von der Adresse adr wird der Wert 8b aus dem Speicher geholt.

cmove (adr1 adr2 u --) 83
"c-move"
Beginnend bei adr1 werden u Bytes zur adr2 kopiert. Zuerst wird das Byte von adr1 nach adr2 bewegt und dann aufsteigend fortgefahren. Wenn u Null ist, wird nichts kopiert.

cmove> (adr1 adr2 u --) 83
"c-move-rauf", engl."c-move-up"
Beginnend bei adr1 werden u Bytes zur adr2 kopiert. Zuerst wird das Byte von adr1-plus-u-minus-1 nach adr2-plus-u-minus-1 bewegt und dann absteigend fortgefahren. Wenn u Null ist, wird nichts kopiert. Das Wort wird benutzt um Speicherinhalte auf hoehere Addressen zu verschieben wenn die Speicherbereiche sich ueberlappen.

count (adr1 -- adr2 +n) 83
adr2 ist adr1+1 und +n ist die Laenge des String. Das Byte mit der Adresse adr1 enthaelt die Laenge des String angegeben in Bytes. Die Zeichen des Strings beginnen bei der Adresse adr+1. Die Laenge +n eines Strings darf im Bereich (0 .. 255) liegen. Vergleiche auch "String, counted".

ctoggle (8b adr --)
"c-toggle"
Fuer jedes gesetzte Bit in 8b wird im Byte mit der Adresse adr das entsprechende Bit invertiert (dh. ein zuvor gesetztes Bit ist danach geloescht und ein zuvor geloeschtes Bit ist danach gesetzt). Fuer alle geloeschten Bits in n bleiben die entsprechenden Bits im Byte mit der Adresse adr unveraendert. Der Ausdruck DUP C@ ROT XOR SWAP C! wirkt genauso.

erase (adr u --)
Von adr an werden u Bytes des Speichers mit $00 ueberschrieben. Hat u den Wert Null, passiert nichts.

fill (adr u 8b --)
Von adr an werden u Bytes des Speichers mit 8b beschrieben. Hat u den Wert Null, passiert nichts.

move (adr1 adr2 u --)
Beginnend bei adr1 werden u Bytes nach adr2 bewegt. Dabei ist es ohne Bedeutung, ob ueberlappende Speicherbereiche aufwaerts oder abwaerts kopiert werden, weil MOVE die passende Routine dazu auswaehlt. Hat u den Wert Null, passiert nichts. Siehe auch CMOVE und CMOVE> .

off (adr --)
Schreibt der Wert FALSE in den Speicher mit der Adresse adr.

on (adr --)
Schreibt der Wert TRUE in den Speicher mit der Adresse adr.

pad (-- adr) 83
adr ist die Startaddresse einer "scratch area". In diesen Speicherbereich koennen Daten fuer Zwischenrechnungen abgelegt werden. Wenn die naechste verfuegbare Stelle fuer das Dictionary veraendert wird, aendert sich auch die Startadresse von pad. Die vorherige Startadresse von pad geht ebenso wie die Daten dort verloren. pad erstreckt sich bis zum obersten Wert des Datenstack.

place (adr1 +n adr2 --)
Bewegt +n Bytes von der Adresse adr1 zur Adresse adr2+1 und schreibt +n in die Speicherstelle mit der Adresse adr2. Wird in der Regel benutzt um Text einer bestimmten Laenge als Counted String abzuspeichern. adr2 darf gleich, groesser und auch kleiner als adr1 sein.

32-Bit-Worte

d0= (d -- flag) 83
"d-null-gleich", engl."d-zero-equals"
Wenn d gleich Null ist, wird flag wahr.

d+ (d1 d2 -- d3) 83
"d-plus"
d1 und d2 addiert ergibt d3.

d< (d1 d2 -- flag) 83
"d-kleiner-als", engl."d-less-than"
Wenn d1 kleiner als d2 ist, wird flag wahr. Benutzt die < Operation, jedoch fuer 32bit Vergleiche.

dabs (d -- ud) 83
"d-absolut"
ud ist der Betrag von d. Wenn d gleich -2.147.483.648 ist, hat ud den selben Wert wie d.

dnegate

(d1 -- d2) 83
"d-negate"
d2 hat den gleichen Betrag, aber ein anderes Vorzeichen als d1.

extend (n -- d)
Der Wert n wird auf den doppelt genauen Wert d vorzeichenrichtig erweitert.

m* (n1 n2 -- d)
"m-mal", engl."m-times"
Der Wert n1 wird mit n2 multipliziert. d ist das doppeltgenaue vorzeichenrichtige Produkt.

m/mod (d n1 -- n2 n3)
"m-durch-mod", engl."m-divide-mod"
n2 ist der Rest und n3 der Quotient aus der Division der doppeltgenauen Zahl d durch den Divisor n1. Der Rest n2 hat dasselbe Vorzeichen wie d oder ist Null. Eine Fehlerbedingung besteht, wenn der Divisor Null ist oder der Quotient ausserhalb des Intervalls (-32768 .. 32767) liegt.

ud/mod (ud1 u1 -- u2 ud2)
"u-d-durch-mod", engl."u-d-divide-mod"
u2 ist der Rest und ud2 der doppeltgenaue Quotient aus der Division der doppeltgenauen Zahl ud1 durch den Divisor u1. Die Zahlen u sind vorzeichenlose 16-Bit-Zahlen (unsigned integer). Eine Fehlerbedingung besteht, wenn der Divisor Null ist.

um* (u1 u2 -- ud) 83
"u-m-mal", engl."u-m-times"
Die Werte u1 und u2 werden multipliziert und als doppeltgenauer Wert d abgelegt. UM* ist die anderen multiplizierenden Worten zugrundeliegende Routine. Die Zahlen u sind vorzeichenlose Zahlen.

um/mod (ud u1 -- u2 u3) 83

"u-m-durch-mod", engl."u-m-divide-mod"
u2 ist der Rest und u3 der Quotient aus der Division von ud durch den Divisor u1. Die Zahlen u sind vorzeichenlose Zahlen. Eine Fehlerbedingung besteht, wenn der Divisor Null ist oder der Quotient ausserhalb des Intervalls (0 .. 65535) liegt.
UM/MOD ist die allen anderen dividierenden Worten zugrundeliegende Routine. Die Fehlermeldung "division overflow" wird ausgegeben, wenn der Divisor Null ist.

Stack

-roll (16bn...16b1 16b0 +n -- 16b0 16bn...16b1)
"minus-roll"
Das oberste Glied einer Kette von n Werten wird an die nte Position gerollt. Dabei wird +n selbst nicht mitgezaehlt. 2 -ROLL wirkt wie -ROT. 0 -ROLL veraendert nichts.

-rot (16b1 16b2 16b3 -- 16b3 16b1 16b2)
"minus-rot"
Die drei obersten 16b Werte werden rotiert, sodass der oberste Wert zum Untersten wird. Hebt ROT auf.

.s (--)
"punkt-s", engl."dot-s"
Gibt alle Werte die auf dem Stack liegen aus, ohne den Stack zu veraendern. Oft benutzt um neue Worte auszutesten. Die Ausgabe der Werte erfolgt von links nach rechts, der oberste Stackwert zuerst.

2dup (32b -- 32b 32b) 83
"zwei-dup", engl."two-dup"
Der Wert 32b wird dupliziert.

2drop (32b --) 83
"two-drop"
Der Wert 32b wird fallengelassen, dh. die beiden obersten Werte werden vom Stack entfernt.

2swap (32b1 32b2 -- 32b2 32b1) 83
"zwei-swap", engl."two-swap"
Die beiden obersten 32b Werte werden vertauscht.

?dup (16b -- 16b 16b)
oder (0 -- 0)
"fragezeichen-dup", engl."question-dup"
Nur wenn der Wert 16b verschieden von Null ist, wird er verdoppelt.

clearstack (-- empty)
Loescht den Datastack, indem die Stack-Startadresse aus der Uservariablen S0 in den Datastackzeiger (stackpointer) geschrieben wird.

depth (-- n)
n ist die Anzahl der Werte die auf dem Stack lagen, bevor DEPTH ausgefuehrt wurde.

drop (16b --) 83
Der Wert 16b wird fallengelassen, dh. der oberste Wert wird vom Stack entfernt. Siehe auch 2DROP.

dup (16b -- 16b 16b) 83
Der Wert 16b wird verdoppelt. Siehe auch 2DUP.

nip (16b1 16b2 -- 16b2)
Der zweite Wert wird vom Stack entfernt. 16b2 ist der oberste und 16b1 der zweite Wert auf dem Stack.

over (16b1 16b2 -- 16b1 16b2 16b1) 83
Der Wert 16b1 wird ueber 16b2 herueber kopiert.

pick (16bn..16b0 +n -- 16bn..16b0 16bn) 83
Der +nte Wert im Stack wird oben auf den Stack kopiert. Dabei wird +n selbst nicht mitgezaehlt. 0 PICK wirkt wie DUP. 1 PICK wirkt wie OVER.

roll (16bn 16bm..16b0 +n -- 16bm..16b0 16bn) 83
Das +nte Glied einer Kette von n Werten wird nach oben auf den Stack gerollt. Dabei wird +n selbst nicht mitgezaehlt.

rot (16b1 16b2 16b3 -- 16b2 16b3 16b1) 83
Die drei obersten 16b Werte werden rotiert, sodass der unterste zum obersten wird.

s0 (-- adr)
adr ist die Adresse einer Uservariablen, in der die Startadresse des Datastack steht. Der Ausdruck S0 @ SP! wirkt wie CLEARSTACK und leert den Stack.

swap (16b1 16b2 -- 16b2 16b1) 83
Die beiden obersten 16b-Werte werden vertauscht.

sp! (adr --)
"s-p-store"
Setzt den Datastackzeiger (engl."stack pointer") auf die Adresse adr. Der oberste Wert im Datastack ist nun der, welcher in der Speicherstelle bei adr steht.

sp@ (-- adr)
"s-p-fetch"
Holt die Adresse adr aus dem Datastackzeiger. Der oberste Wert im Stack stand in der Speicherstelle bei adr, bevor SP! ausgefuehrt wurde. Der Ausdruck SP@ @ wirkt wie DUP.

under (16b1 16b2 -- 16b2 16b1 16b2)
Eine Kopie des obersten Wertes auf dem Stack wird unter den zweiten Wert eingefuegt. Der Ausdruck SWAP OVER wirkt genauso.

Returnstack

>r (16b --) C,83
"auf-r", engl."to-r"
Der Wert 16b wird auf den Returnstack gelegt. Siehe auch R> .

push (adr --)
Der Inhalt aus der Adresse adr wird auf dem Returnstack bis zum naechsten EXIT verwahrt und sodann nach adr zurueck geschrieben. Dies ermoeglicht die lokale Verwendung von Variablen innerhalb einer :-Definition. Wird zB. benutzt in der Form:
: WORT ... BASE PUSH HEX ... ;
Hier wird innerhalb von WORT in der Zahlenbasis HEX gearbeitet, zB. um ein Speicherbereich auszugeben (Hex-Dump). Das Wort ; fuehrt EXIT aus. Nachdem WORT ausgefuehrt worden ist, besteht die gleiche Zahlenbasis wie vorher.

r> (-- 16b) C,83
"r-von", engl."r-from"
Der Wert 16b wird von dem Returnstack geholt. Siehe auch >R .

rp! (adr --)
"r-p-store"
Setzt den Returnstackzeiger (engl."return stack pointer") auf die Adresse adr. Der oberste Wert im Returnstack ist nun der, welcher in der Speicherstelle bei adr steht.

r0 (-- adr)
adr ist die Adresse einer Uservariablen, in der die Startadresse des Returnstack steht.

r@ (-- 16b) C,83
"r-fetch"
Der Wert 16b ist eine Kopie des obersten Wertes im Returnstack.

rdepth (-- n)
"r-depth"
n ist die Anzahl der Werte die auf dem Returnstack liegen.

rdrop (--) C
"r-drop"
Der Wert 16b wird vom Returnstack fallengelassen, dh. der oberste Wert wird vom Returnstack entfernt. Der Datenstack wird nicht veraendert.

rp@ (-- adr)
"r-p-fetch"
Holt die Adresse adr aus dem Returnstackzeiger. Der oberste Wert im Returnstack steht in der Speicherstelle bei adr.

Strings

" (-- adr) C,I
(text (--) compiling
"string"
Liest den Text bis zum naechsten " und legt ihn als Counted String im Dictionary ab. Kann nur bei der Kompilation verwendet werden. Zur Laufzeit wird Startadresse des Counted String auf den Stack gelegt. Benutzt in der Form:
: <name> ... " <text>" ... ;

(+d1 -- +d2) 83
(1), engl."sharp";
Der Rest von +d1 geteilt durch den Wert in BASE wird in ein ASCII Zeichen umgewandelt und dem Ausgabestring in Richtung absteigender Addressen hinzugefuegt. +d2 ist der Quotient und verbleibt auf dem Stack zur weiteren Bearbeitung. Ueblicherweise benutzt zwischen <# und #> .
((1) Bisher gibt es keine verbindliche deutsche Aussprache. Das Zeichen wird in der Notenschrift verwendet und dort in engl."sharp" und in deutsch "kreutz" ausgesprochen. # wird auch oft statt Nr. verwendet; zB. SCR#)

#> (32b -- adr +n) 83
"sharp-greater"
Am Ende der strukturierten Zahlenausgebe wird der 32b Wert vom Stack entfernt. Hinterlegt werden die Adresse des erzeugten Ausgabestrings und +n als die Anzahl Zeichen im Ausgabestring, passend fuer TYPE.

#s (+d -- 0 0) 83
"sharp-s"
+d wird umgewandelt bis der Quotient zu Null geworden ist. Siehe auch # . Dabei wird jedes Zwischenergebnis in ein Ascii-Zeichen umgewandelt und dem String fuer die strukturierte Zahlenausgabe angefuegt. Wenn die Zahl von vorn herein den Wert Null hatte, wird eine einzelne Null in den String gegeben. Wird ueblicherweise benutzt zwischen <# und #> .

/string (adr1 n1 n2 -- adr2 n3)
n2 ist ein Index, welcher in den String weist, der im Speicher bei der Adresse adr1 beginnt. Der String hat die Laenge n1. n3 ist gleich n1-minus-n2, wenn n2 kleiner als n1 ist, und ad2 ist dann adr1-plus-n2. n3 ist gleich Null, wenn n2 groesser oder gleich n1 ist, und adr2 ist dann adr1-plus-n1. Der Vergleich benutzt die Operation U< . Wird zB. benutzt, um die vorderen n Zeichen eines String abzuschneiden.

<# (--) 83
"less-sharp"
Leitet die strukturierte Zahlenausgabe ein. Um eine doppelt genaue Zahl in einen von rechts nach links aufgebauten Ascii-String umzuwandeln, benutze man die Worte:
#> #s <# HOLD SIGN

accumulate (+d1 adr char -- +d2 adr)
Dient der Umwandlung von Ziffern in Zahlen. Multipliziert die Zahl +d1 mit BASE, um sie eine Stelle in der aktuellen Zahlenbasis nach links zu ruecken, und addiert den Zahlenwert von char dazu. char stellt eine Ziffer dar. adr wird nicht veraendert. Wird zB. in CONVERT benutzt. +n muss eine in der Zahlenbasis BASE gueltige Ziffer darstellen.

capital (char1 -- char2)
Die Zeichen im Bereich von a bis z werden in die Grossbuchstaben A bis Z umgewandelt. Andere Zeichen werden nicht veraendert.

capitalize (adr -- adr)
Wandelt alle Klein-Buchstaben im Counted String bei der Adresse adr in Gross-Buchstaben um. adr wird nicht veraendert. Siehe auch CAPITAL.

convert (+d1 adr1 -- +d2 adr2) 83
Wandelt den Ascii-Text ab adr1+1 in eine Zahl mit der Zahlenbasis BASE um. Der entstehende Wert wird in d1 akkumuliert und als d2 hinterlassen. adr2 ist die Adresse des ersten nicht umwandelbaren Zeichens im Text.

digit? (char -- digit true)
oder (char -- false)
Prueft mit BASE ob das Zeichen char eine gueltige Ziffer ist. Ist diese wahr, wird der Zahlenwert der Ziffer und TRUE auf dem Stack hinterlegt. Ist char keine gueltige Ziffer, wird FALSE hinterlegt.

hold (char --) 83
Das Zeichen char wird in den bildhaften Ausgabestring eingefuegt. Ueblicherweise benutzt zwischen <# und #> .

nullstring? (adr -- adr false)
oder (adr -- true)
Prueft, ob der Counted String bei der Adresse adr ein String der Laenge Null ist. Wenn dies der Fall ist, wird TRUE hinterlegt. Sonst bleibt adr erhalten und FALSE wird obenauf gelegt.

number (adr -- d)
Wandelt den Counted String mit der Adresse adr in die Zahl d um. Die Umwandlung erfolgt entsprechend der Zahlenbasis in BASE. Eine Fehlerbedingung besteht, wenn die Ziffern des String nicht in eine Zahl verwandelt werden koennen.

number? (adr -- d Ø>)
oder (adr -- n Ø<)
oder (adr -- adr false)
Wandelt den Counted String mit der Adresse adr in die Zahl n um. Die Umwandlung erfolgt entsprechend der Zahlenbasis in BASE oder wird vom ersten Zeichen im String bestimmt. Enthaelt der String zwischen den Ziffern auch Ascii-Zeichen fuer Komma oder Punkt, so wird er als doppelt-genaue Zahl d interpretiert. Sonst wird der String in eine einfach-genaue Zahl n umgewandelt. Wenn die Ziffern des String nicht in eine Zahl verwandelt werden koennen, bleibt die Adresse des String erhalten und der Wert fuer logisch-falsch wird auf den Stack gelegt. Die Zeichen, die zur Bestimmung der Zahlenbasis dem Ziffernstring vorangestellt werden koennen, sind:
% (Basis 2 "binaer")
& (Basis 10 "decimal")
$ (Basis 16 "hexadecimal")
h (Basis 16 "hexadecimal")
Der Wert in BASE wird dadurch nicht veraendert.

scan (adr1 n1 char -- adr2 n2)
Der String mit der Laenge n1, der im Speicher ab Adresse adr1 steht, wird nach dem Zeichen char durchsucht. adr2 ist die Adresse, bei der das Zeichen char gefunden wurde. n2 ist die Laenge des verbleibenden String. Wird char nicht gefunden, ist adr2 die Adresse des ersten Zeichen hinter dem String und n2 ist Null.

sign (n --) 83
Wenn n negativ ist, wird ein Minuszeichen in den bildhaften Ausgabestring eingefuegt. Wird ueblicherweise benutzt zwischen <# und #> .

skip (adr1 n1 char -- adr2 n2)
Der String mit der Laenge n1, der im Speicher ab Adresse adr1 steht, wird nach dem ersten Zeichen, das verschieden von char ist, durchsucht. adr2 ist die Adresse dieses Zeichens. n2 ist die Laenge des verbleibenden String. Besteht der gesammte String aus den Zeichen char, ist adr2 die Adresse des ersten Zeichen hinter dem String und n2 ist Null.

Datentypen

: (-- sys) 83 "colon"
ein definierendes Wort, das in der Form:
: <name> ... ;
benutzt wird. Es erzeugt die Wortdefinition fuer <name> im Kompilations- Vokabular und schaltet den Kompiler an. Das erste Vokabular in der Suchreihenfolge wird durch das Kompilations- Vokabular ersetzt. Das Kompilations- Vokabular wird nicht geaendert. Der Quelltext wird anschliessend kompiliert. <name> wird eine "colon-definition" oder eine ":-Definition" genannt. Die neue Wortdefinition fuer <name> kann solange nicht im Dictionary gefunden werden, bis das zugehoerige ; oder ;CODE erfolgreich ausgefuehrt wurde.

Eine Fehlerbedingung liegt vor, wenn ein Wort nicht gefunden und nicht in eine Zahl gewandelt werden kann oder wenn, waehrend der Kompilation vom Massenspeicher, der Quelltext erschoepft ist, bevor ; oder ;CODE erreicht werden. sys wird vom zugehoerigen ; abgebaut.

; (--) 83 I C "semi-colon"
(sys --) compiling
beendet die Kompilation einer :-Definition; macht den Namen <name> dieser :-Definition im Dictionary auffindbar, schaltet den Interpreter ein und kompiliert ein EXIT. Die Stackparameter sys, die von : hinterlassen wurden, werden geprueft und abgebaut. Siehe : und EXIT .

Alias (cfa --)
ein definierendes Wort, das typisch in der Form:
' <namex> Alias <name>
benutzt wird. ALIAS erzeugt einen Kopf fuer <name> im Dictionary. Wird <name> aufgerufen, so verhaelt es sich wie <namex>. Insbesondere wird beim Kompilieren nicht <namex>, sondern <name> ins Dictionary eingetragen. Im Unterschied zu
: <name> <namex> ;
ist es mit ALIAS auch moeglich, Worte, die den Returnstack beeinflussen (vergleiche >R oder R>), mit anderem Namen zu definieren. Ausser dem neuen Kopf fuer <name> wird kein zusaetzlicher Speicherplatz verbraucht.

Constant (16b --) 83
ein definierendes Wort, das in der Form:
16b Constant <name>
benutzt wird. Wird <name> spaeter ausgefuehrt, so wird 16b auf den Stack gelegt.

Defer (--)
ein definierendes Wort, das in der Form:
Defer <name>
benutzt wird. Erzeugt den Kopf fuer ein neues Wort <name> im Dictionary, haelt 2 Bytes in dessen Parameterfeld frei und speichert dort die Kompilationsadresse einer Fehlerroutine. Wird <name> nun ausgefuehrt, so wird die Ausfuehrung mit einer Fehlermeldung abgebrochen. Man kann dem Wort <name> zu jedem Zeitpunkt eine andere Funktion zuweisen, siehe IS . <name> laesst sich jedoch schon kompilieren, ohne dass es eine sinnvolle Aktion ausfuehrt. Damit ist die Kompilation erst spaeter definierter Worte indirekt moeglich. Andererseits ist die Veraenderung des Verhaltens von <name> fuer spezielle Zwecke moeglich.

Deferred Worte im System sind:
R/W , 'COLD , 'RESTART , 'ABORT , 'QUIT ,
NOTFOUND , .STATUS und DISKERR .
Ein spezielles Deferred Wort ist
>INTERPRET.

Input: (--) "input-colon"
ein definierendes Wort, benutzt in der Form:
Input: <name>
newKEY newKEY? newDECODE newEXPECT [
INPUT: erzeugt einen Kopf fuer <name> im Dictionary und kompiliert einen Vektor von Zeigern auf Worte, die fuer die Eingabe von Zeichen zustaendig sind. Wird <name> ausgefuehrt, so wird ein Zeiger auf das Parameterfeld von <name> in die Uservariable INPUT geschrieben. Alle Eingaben werden jetzt ueber die neuen Eingabeworte abgewickelt. Die Reihenfolge der Worte nach INPUT: <name> bis zur [muss eingehalten werden.

Im System ist das mit INPUT: definierte Wort KEYBOARD enthalten, wenn der Editor geladen ist, stehen ausserdem EDIBOARD und DIGITS zur Verfuegung.

Is (cfa --)
ein Wort, mit dem das Verhalten eines Deferred Wortes veraendert werden kann. IS wird in der Form:
' <namex> Is <name>
benutzt. Wenn <name> kein Deferred Wort ist, so wird eine Fehlerbehandlung eingeleitet, sonst wird <name> die Ausfuehrung von <namex> zugewiesen. Siehe DEFER.

Output: (--) "output-colon"
ein definierendes Wort, benutzt in der Form:
Output: <name>
newEMIT newCR newTYPE newDEL newPAGE newAT
newAT? [
OUTPUT: erzeugt einen Kopf fuer <name> im Dictionary und kompiliert einen Vektor aus Zeigern auf Worte, die fuer die Ausgabe von Zeichen verantwortlich sind. Wird <name> ausgefuehrt, so wird ein Zeiger auf das Parameterfeld von <name> in die Uservariable OUTPUT geschrieben. Alle Ausgaben werden jetzt ueber die neuen Ausgabeworte abgewickelt. Die Reihenfolge der Worte nach OUTPUT: <name> bis zur [muss eingehalten werden.

Im System ist das mit OUTPUT: definierte Wort DISPLAY enthalten.

User (--) 83
ein definierendes Wort, benutzt in der Form:
User <name>
USER erzeugt einen Kopf fuer <name> und haelt 2 Byte in der Userarea frei. Siehe UALLOT . Diese 2 Byte werden fuer den Inhalt der USERvariablen benutzt und werden nicht initialisiert. Im Parameterfeld der USERvariablen im Dictionary wird nur ein Byte- Offset zum Beginn der Userarea abgelegt. Wird <name> spaeter ausgefuehrt, so wird die Adresse des Wertes der USERvariablen in der Userarea auf den Stack gegeben.

Variable (--) 83
ein definierendes Wort, benutzt in der Form:
Variable <name>
VARIABLE erzeugt einen Kopf fuer <name> und haelt 2 Byte in seinem Parameterfeld frei. Siehe ALLOT. Dies Parameterfeld wird fuer den Inhalt der VARIABLEn benutzt und wird nicht initialisiert. Wird <name> spaeter ausgefuehrt, so wird die Adresse des Parameterfeldes auf den Stack gegeben.

Vocabulary (--) 83
ein definierendes Wort, das in der Form:
Vocabulary <name>
benutzt wird. VOCABULARY erzeugt einen Kopf fuer <name>, das den Anfang einer neuen Liste von Worten bildet. Spaetere Ausfuehrung von <name> ersetzt das erste Vokabular in der Suchreihenfolge durch <name>. Wird <name> das Kompilations Vokabular, so werden neue Definitionen in die Liste von <name> gelinkt. Vergleiche DEFINITIONS .

Dictionary - Worte

' (-- addr) 83 "tick"
Wird in der Form ' <name> benutzt.
addr ist die Kompilationsadresse von <name>. Wird <name> nicht in der Suchreihenfolge gefunden, so wird eine Fehlerbehandlung eingeleitet.

(forget (addr --) "paren-forget"
Entfernt alle Worte, deren Kompilationsadresse oberhalb von addr liegt, aus dem Dictionary und setzt HERE auf addr. Ein Fehler liegt vor, falls addr im Heap liegt.

, (16b --) 83 "comma"
2 ALLOT fuer 16b und speichere 16b ab HERE 2- .

.name (addr --) "dot-name"
addr ist die Adresse des Countfeldes eines Namens. Dieser Name wird ausgedruckt. Befindet er sich im Heap, so wird das Zeichen ¦ vorangestellt. Ist addr null, so wird "???" ausgegeben.

allot (w --) 83
Allokiere w Bytes im Dictionary. Die Adresse des naechsten freien Dictionaryplatzes wird entsprechend verstellt.

c, (16b --) "c-comma"
ALLOT ein Byte und speichere die unteren 8 Bit von 16b in HERE 1-.

clear (--)
Loescht alle Namen und Worte im Heap.

dp (-- addr) "d-p"
Eine Uservariable, die die Adresse des naechsten freien Dictionaryplatzes enthaelt.

empty (--)
Loescht alle Worte, die nach der letzten Ausfuehrung von SAVE oder dem letzten Kaltstart definiert wurden. DP wird auf seinen Kaltstartwert gesetzt und der Heap geloescht.

forget (--) 83
Wird in der Form FORGET <name> benutzt
Falls <name> in der Suchreihenfolge gefunden wird, so werden <name> und alle danach definierten Worte aus dem Dictionary entfernt. Wird <name> nicht gefunden, so wird eine Fehlerbehandlung eingeleitet. Liegt <name> in dem durch SAVE geschuetzten Bereich, so wird ebenfalls eine Fehlerbehandlung eingeleitet. Es wurden Vorkehrungen getroffen, die es ermoeglichen, aktive Tasks und Vokabulare, die in der Suchreihenfolge auftreten, zu vergessen.

here (-- addr) 83
addr ist die Adresse des naechsten freien Dictionaryplatzes.

hide (--)
Entfernt das zuletzt definierte Wort aus der Liste des Vokabulars, in das es eingetragen wurde. Dadurch kann es nicht gefunden werden. Es ist aber noch im Speicher vorhanden. (s.a. REVEAL LAST)

last (-- addr)
Variable, die auf das Countfeld des zuletzt definierten Wortes zeigt.

name> (addr1 -- addr2) "name-from"
addr2 ist die Kompilationsadresse die mit dem Countfeld in addr1 korrespondiert.

origin (-- addr)
addr ist die Adresse, ab der die Kaltstartwerte der Uservariablen abgespeichert sind.

reveal (--)
Traegt das zuletzt definierte Wort in die Liste des Vokabulars ein, in dem es definiert wurde.

save (--)
Kopiert den Wert aller Uservariablen in den Speicherbereich ab ORIGIN und sichert alle Vokabularlisten. Wird spaeter COLD ausgefuehrt, so befindet sich das System im gleichen Speicherzustand wie bei Ausfuehrung von SAVE.

uallot (n1 -- n2)
Allokiere bzw. deallokiere n1 Bytes in der Userarea. n2 gibt den Anfang des allokierten Bereiches relativ zum Beginn der Userarea an. Eine Fehlerbehandlung wird eingeleitet, wenn die Userarea voll ist.

udp (-- addr) "u-d-p"
Eine Uservariable, in dem das Ende der bisher allokierten Userarea vermerkt ist.

>body (addr1 -- addr2) "to-body"
addr2 ist die Parameterfeldadresse, die mit der Kompilationsadresse addr1 korrespondiert.

>name (addr1 -- addr2) "to-name"
addr2 ist die Adresse eines Countfeldes, das mit der Kompilationsadresse addr1 korrespondiert. Es ist moeglich, dass es mehrere addr2 fuer ein addr1 gibt. In diesem Fall ist nicht definiert, welche ausgewaehlt wird.

Vokabular - Worte

also (--)
Ein Wort, um die Suchreihenfolge zu spezifizieren. Das Vokabular im auswechselbarem Teil der Suchreihenfolge wird zum ersten Vokabular im festen Teil gemacht, wobei die anderen Vokabulare des festen Teils nach hinten ruecken. Ein Fehler liegt vor, falls der feste Teil sechs Vokabulare enthaelt.

Assembler (--)
Ein Vokabular, das Prozessor-spezifische Worte enthaelt, die fuer Code-Definitionen benoetigt werden.

context (-- addr)
addr ist die Adresse des auswechselbaren Teils der Suchreihenfolge. Sie enthaelt einen Zeiger auf das erste zu durchsuchende Vokabular.

current (-- addr)
addr ist die Adresse eines Zeigers, der auf das Kompilationsvokabular zeigt, in das neue Worte eingefuegt werden.

definitions (--) 83
Ersetzt das gegenwaertige Kompilationsvokabular durch das Vokabular im auswechselbaren Teil der Suchreihenfolge, d.h. neue Worte werden in dieses Vokabular eingefuegt.

Forth (--) 83
Das urspruengliche Vokabular.

forth-83 (--) 83
Lt. Forth83-Standard soll dieses Wort sicherstellen, dass ein Standardsystem benutzt wird. Im ultraFORTH funktionslos.

Only (--)
Loescht die Suchreihenfolge vollstaendig und ersetzt sie durch das Vokabular ONLY im festen und auswechselbaren Teil der Suchreihenfolge. ONLY enthaelt nur wenige Worte, die fuer die Erzeugung einer Suchreihenfolge benoetigt werden.

Onlyforth (--)
entspricht der haeufig benoetigten Sequenz
ONLY FORTH ALSO DEFINITIONS.

seal (--)
Loescht das Vokabular ONLY , so dass es nicht mehr durchsucht wird. Dadurch ist es moeglich, nur die Vokabulare des Anwenderprogramms durchsuchen zu lassen.

toss (--)
Entfernt das erste Vokabular des festen Teils der Suchreihenfolge. Insofern ist es das Gegenstueck zu ALSO .

words (--)
Gibt die Namen der Worte des Vokabulars, das im auswechselbaren Teil der Suchreihenfolge steht, aus, beginnend mit dem zuletzt erzeugtem Namen.

voc-link (-- addr)
Eine Uservariable. Sie enthaelt den Anfang einer Liste mit allen Vokabularen. Diese Liste wird u.a. fuer FORGET benoetigt.

vp (-- addr) "v-p"
Eine Variable, die das Ende der Suchreihenfolge markiert. Sie enthaelt ausserdem Informationen ueber die Laenge der Suchreihenfolge.

Heap - Worte

?head (-- addr) "question-head"
Eine Variable, die angibt, ob und wieviele der naechsten zu erzeugenden Namen im Heap angelegt werden sollen (s.a. ¦).

hallot (n --)
Allokiere bzw. deallokiere n Bytes auf dem Heap. Dabei wird der Stack verschoben, ebenso wie der Beginn des Heap.

heap (-- addr)
addr ist der Anfang des Heap. Er wird u.A. durch HALLOT geaendert.

heap? (addr -- flag) "heap-question"
Das Flag ist wahr, wenn addr ein Byte im Heap adressiert, ansonsten falsch.

¦ (--) "headerless"
Setzt bei Ausfuehrung ?HEAD so, dass der naechste erzeugte Name nicht im normalen Dictionaryspeicher angelegt wird, sondern auf dem Heap.

Kontrollstrukturen

+LOOP (n --) 83,I,C "plus-loop"
(sys --) compiling
n wird zum Loopindex addiert. Falls durch die Addition die Grenze zwischen limit-1 und limit ueberschritten wurde, so wird die Schleife beendet und die Loop-Parameter werden entfernt. Wurde die Schleife nicht beendet, so wird sie hinter dem korrespondierenden DO bzw. ?DO fortgesetzt.

?DO (w1 w2 --) 83,I,C "question-do"
(-- sys) compiling
Wird in der folgenden Art benutzt:
?DO ... LOOP bzw. ?DO ... +LOOP
Beginnt eine Schleife. Der Schleifenindex beginnt mit w2, limit ist w1. Details ueber die Beendigung von Schleifen: s. +LOOP. Ist w2=w1 , so wird der Schleifenrumpf ueberhaupt nicht durchlaufen.

?exit (flag --) "question-exit"
Fuehrt EXIT aus, falls das flag wahr ist. Ist das flag falsch, so geschieht nichts.

BEGIN (--) 83,I,C
(sys ---) compiling
Wird in der folgenden Art benutzt:
BEGIN (...flag WHILE) ... flag UNTIL
oder: BEGIN (...flag WHILE) ... REPEAT
BEGIN markiert den Anfang einer Schleife. Der ()-Ausdruck ist optional und kann beliebig oft auftreten. Die Schleife wird wiederholt, bis das flag bei UNTIL wahr oder oder das flag bei WHILE falsch ist. REPEAT setzt die Schleife immer fort.

bounds (start count -- limit start)
Dient dazu, ein Intervall, das durch Anfangswert start und Laenge count gegeben ist, in ein Intervall umzurechnen, das durch Anfangswert start und Endwert+1 limit beschrieben wird.

Beispiel : 10 3 bounds DO ... LOOP fuehrt dazu, das I die Werte 10 11 12 annimmt.

DO (w1 w2 --) 83,I,C
(sys --) compiling
Entspricht ?DO , jedoch wird der Schleifenrumpf mindestens einmal durchlaufen. Ist w1=w2 , so wird der Schleiferumpf 65536-mal durchlaufen.

ELSE (--) 83,I,C
(sys1 -- sys2)compiling
Wird in der folgenden Art benutzt:
flag IF ... ELSE ... THEN
ELSE wird unmittelbar nach dem Wahr-Teil , der auf IF folgt, ausgefuehrt. ELSE setzt die Ausfuehrung unmittelbar hinter THEN fort.

execute (addr --) 83
Das Wort, dessen Kompilationsadresse addr ist, wird ausgefuehrt.

I (-- w) 83,C
Wird zwischen DO und LOOP benutzt, um eine Kopie des Schleifenindex auf den Stack zu holen.

IF (flag --) 83,I,C
(-- sys) compiling
Wird in der folgenden Art benutzt:
flag IF ... ELSE ... THEN
oder: flag IF ... THEN
Ist das flag wahr, so werden die Worte zwischen IF und ELSE ausgefuehrt und die Worte zwischen ELSE und THEN ignoriert. Der ELSE-Teil ist optional.
Ist das flag falsch, so werden die Worte zwischen IF und ELSE (bzw. zwischen IF und THEN , falls ELSE nicht vorhanden ist) ignoriert.

J (-- w) 83,C
Wird zwischen DO .. DO und LOOP .. LOOP benutzt, um eine Kopie des Schleifenindex der aeusseren Schleife auf den Stack zu holen.

LEAVE (--) 83,C
Setzt die Ausfuehrung des Programmes hinter dem naechsten LOOP oder +LOOP fort, wobei die zugehoehrige Schleife beendet wird. Mehr als ein LEAVE pro Schleife ist moeglich, ferner kann LEAVE zwischen anderen Kontrollstrukturen auftreten. Der Forth83-Standard schreibt abweichend vom ultraFORTH vor, dass LEAVE ein immediate Wort ist.

LOOP (--) 83,I,C
(-- sys) compiling
Entspricht +LOOP, jedoch mit n=1 fest gewaehlt.

perform (addr --)
addr ist eine Adresse, unter der sich ein Zeiger auf die Kompilationsadresse eines Wortes befindet. Dieses Wort wird ausgefuehrt. Entspricht der Sequenz @ EXECUTE .

REPEAT (--) 83,I,C
(-- sys) compiling
Wird in der folgenden Form benutzt:
BEGIN (.. WHILE) .. REPEAT
REPEAT setzt die Ausfuehrung der Schleife unmittelbar hinter BEGIN fort. Der ()-Ausdruck ist optional und kann beliebig oft auftreten.

THEN (--) 83,I,C
(sys --) compiling
Wird in der folgenden Art benutzt:
IF (...ELSE) ... THEN
Hinter THEN ist die Programmverzweigung zuende.

UNTIL (flag --) 83,I,C
(sys --) compiling
Wird in der folgenden Art benutzt:
BEGIN (... flag WHILE) ... flag UNTIL
Markiert das Ende einer Schleife, deren Abbruch durch flag herbeigefuehrt wird. Ist das flag vor UNTIL wahr, so wird die Schleife beendet, ist es falsch, so wird die Schleife unmittelbar hinter BEGIN fortgesetzt.

WHILE (flag --) 83,I,C
(sys1 -- sys2)compiling
Wird in der folgenden Art benutzt:
BEGIN .. flag WHILE .. REPEAT
oder: BEGIN .. flag WHILE .. flag UNTIL
Ist das flag vor WHILE wahr, so wird die Ausfuehrung der Schleife bis UNTIL oder REPEAT fortgesetzt, ist es falsch, so wird die Schleife beendet und das Programm hinter UNTIL bzw. REPEAT fortgesetzt. Es koennen mehre WHILE in einer Schleife verwendet werden.

Compiler - Worte

," (--) "comma-quote"
Speichert einen counted String im Dictionary ab HERE. Dabei wird die Laenge des Strings in dessen erstem Byte, das nicht zur Laenge hinzugezaehlt wird, vermerkt.

Ascii (-- char) I
(--) compiling
Wird in der folgenden Art benutzt:
Ascii ccc
wobei ccc durch ein Leerzeichen beendet wird. char ist der Wert des ersten Zeichens von ccc im benutzten Zeichensatz (gewoehnlich ASCII). Falls sich das System im kompilierenden Zustand befindet, so wird char als Konstante compiliert. Wird die :-definition spaeter ausgefuehrt, so liegt char auf dem Stack.

compile (--) 83,C
Typischerweise in der folgenden Art benutzt:
: <name> ... compile <namex> ... ;
Wird <name> ausgefuehrt, so wird die Kompilationsadresse von <namex> zum Dictionary hinzugefuegt und nicht ausgefuehrt. Typisch ist <name> immediate und <namex> nicht immediate.

Does> (-- addr) 83,I,C "does"
(--) compiling
Definiert das Verhalten des Wortes, das durch ein definierendes Wort erzeugt wurde. Wird in der folgenden Art benutzt:
: <namex> ... <create> ... Does> ... ;
und spaeter:
<namex> <name>
wobei <create> CREATE oder ein anderes Wort ist, das CREATE ausfuehrt.

Zeigt das Ende des Wort-erzeugenden Teils des definierenden Wortes an. Beginnt die Kompilation des Codes, der ausgefuehrt wird, wenn <name> aufgerufen wird. In diesem Fall ist addr die Parameterfeldadresse von <name>. addr wird auf den Stack gebracht und die Sequenz zwischen DOES> und ; wird ausgefuehrt.

immediate (--) 83
Markiert das zuletzt definierte Wort als "immediate", d.h. dieses Wort wird auch im kompilierenden Zustand ausgefuehrt.

Literal (-- 16b) 83,I,C
(16b --) compiling
Typisch in der folgenden Art benutzt:
[16b] Literal
Kompiliert ein systemabhaengiges Wort, so dass bei Ausuehrung 16b auf den Stack gebracht wird.

recursive (--) I,C
(--) compiling
Erlaubt die rekursive Kompilation des gerade definierten Wortes in diesem Wort selbst. Ferner kann Code fuer Fehlerkontrolle erzeugt werden.

restrict (--)
Markiert das zuletzt definierte Wort als "restrict", d.h. dieses Wort kann nicht vom Textinterpreter interpretiert sondern ausschliesslich in anderen Worten kompiliert werden.

[(--) 83,I "left-bracket"
(--) compiling
Schaltet den interpretierenden Zustand ein. Der Quelltext wird sukzessive ausgefuehrt. Typische Benutzung : s. LITERAL

['] (-- addr) 83,I,C "bracket-tick"
(--) compiling
Wird in der folgenden Art benutzt:
['] <name>
Kompiliert die Kompilationsadresse von <name> als eine Konstante. Wenn die :-definition spaeter ausgefuehrt wird, so wird addr auf den Stack gebracht. Ein Fehler tritt auf, wenn <name> in der Suchreihenfolge nicht gefunden wird.

[compile] (--) 83,I,C "bracket-compile"
(--) compiling
Wird in der folgenden Art benutzt:
[compile] <name>
Erzwingt die Kompilation des folgenden Wortes <name>. Damit ist die Kompilation von immediate-Worten moeglich.

Interpreter - Worte

((--) 83,I "paren"
(--) compiling
Wird in der folgenden Art benutzt:
(ccc)
Die Zeichen ccc, abgeschlossen durch) , werden als Kommentar betrachtet. Kommentare werden ignoriert. Das Leerzeichen zwischen (und ccc ist nicht Teil des Kommentars. (kann im interpretierenden oder kompilierenden Zustand benutzt werden. Fehlt) , so werden alle Zeichen im Quelltext als Kommentar betrachtet.

+load (n --) "plus-load"
LOAD den Block, dessen Nummer um n hoeher ist, als die Nummer des gegenwaertig interpretierten Blockes.

+thru (n1 n2 --) "plus-thru"
LOAD die Bloecke hintereinander, die n1..n2 vom gegenwaertigen Block entfernt sind.
Beispiel : 1 2 +thru laedt die naechsten beiden Bloecke.

--> (--) I "next-block"
(--) compiling
Setze die Interpretation auf dem naechsten Block fort.

>in (-- addr) 83 "to-in"
Eine Variable, die den Offset auf das gegenwaertige Zeichen im Quelltext enthaelt. s. WORD

>interpret (--) "to-interpret"
Ein deferred Wort, das die gegenwaertige Interpretationsroutine aufruft, ohne eine Rueckkehradresse auf dem Returnstack zu hinterlassen (was INTERPRET tut). Es kann als spezielles GOTO angesehen werden.

blk (-- addr) 83 "b-l-k"
Eine Variable, die die Nummer des gerade als Quelltext interpretierten Blockes enthaelt. Ist der Wert von BLK Null, so wird der Quelltext vom Texteingabepuffer genommen.

find (addr1 -- addr2 n)83
addr1 ist die Adresse eines counted string. Der String enthaelt einen Namen, der in der aktuellen Suchreihenfolge gesucht wird. Wird das Wort nicht gefunden, so ist addr2 = addr1 und n = Null. Wird das Wort gefunden, so ist addr2 dessen Kompilationsadresse und n erfuellt folgende Bedingungen:
n ist vom Betrag 2 , falls das Wort restrict ist, sonst vom Betrag 1
n ist positiv, wenn das Wort immediate ist, sonst negativ.

interpret (--)
Beginnt die Interpretation des Quelltextes bei dem Zeichen, das durch den Inhalt von >IN indiziert wird. >IN indiziert relativ zum Anfang des Blockes, dessen Nummer in BLK steht. Ist BLK Null, so werden Zeichen aus dem Texteingabepuffer interpretiert.

load (n --) 83
Die Inhalte von >IN und BLK, die den gegenwaertigen Quelltext angeben, werden gespeichert. Der Block mit der Nummer n wird dann zum Quelltext gemacht. Der Block wird interpretiert. Die Interpretation wird bei Ende des Blocks abgebrochen, sofern das nicht explizit geschieht. Dann wird der alte Inhalt nach BLK und >IN zurueckgebracht. s.a. BLK >IN BLOCK

name (-- addr)
Holt den naechsten String, der durch Leerzeichen eingeschlossen wird, aus dem Quelltext, wandelt ihn in Grossbuchstaben um und hinterlaesst die Adresse addr, ab der der String im Speicher steht. s. WORD

notfound (addr --)
Ein deferred Wort, das aufgerufen wird, wenn der Text aus dem Quelltext weder als Name in der Suchreihenfolge gefunden wurde, noch als Zahl interpretiert werden kann. Kann benutzt werden, um eigene Zahl- oder Stringeingabeformate zu erkennen. Ist mit NO.EXTENSIONS vorbesetzt. Dieses Wort bricht die Interpretation des Quelltextes ab und druckt die Fehlermeldung "haeh?" aus.

parse (char -- addr +n)
Liefert die Adresse addr und Laenge +n des naechsten Strings im Quelltext, der durch den Delimiter char abgeschlossen wird. +n ist Null, falls der Quelltext erschoepft oder das erste Zeichen char ist. >IN wird veraendert.

quit (--) 83
Entleert den Returnstack, schaltet den interpreterierenden Zustand ein, akzeptiert Eingaben von der aktuellen Eingabeeinheit und beginnt die Interpretation des eingegebenen Textes.

source (-- addr +n)
Liefert Anfang addr und maximale Laenge +n des Quelltextes. Ist BLK Null, beziehen sich Anfang und Laenge auf den Texteingabepuffer, sonst auf den Block, dessen Nummer in BLK steht und der in den Rechnerspeicher kopiert wurde. s. BLOCK >IN

state (-- addr) 83
Eine Variable, die den gegenwaertigen Zustand enthaelt. Der Wert Null zeigt den interpretierenden Zustand an, ein von Null verschiedener Wert den kompilierenden Zustand.

thru (n1 n2 --)
LOAD die Bloecke von n1 bis inklusive n2.

word (char -- addr)83
erzeugt einen counted String durch Lesen von Zeichen vom Quelltext, bis dieser erschoepft ist oder der Delimiter char auftritt. Der Quelltext wird nicht zerstoert. Fuehrende Delimiter werden ignoriert. Der gesamte String wird im Speicher beginnend ab Adresse addr als eine Sequenz von Bytes abgelegt. Das erste Byte enthaelt die Laenge des Strings (0..255). Der String wird durch ein Leerzeichen beendet, das nicht in der Laengenangabe enthalten ist. Ist der String laenger als 255 Zeichen, so ist die Laenge undefiniert. War der Quelltext schon erschoepft, als WORD aufgerufen wurde, so wird ein String der Laenge Null erzeugt.
Wird der Delimiter nicht im Quelltext gefunden, so ist der Wert von >IN die Laenge des Quelltextes. Wird der Delimiter gefunden, so wird >IN so veraendert, dass >IN das Zeichen hinter dem Delimiter indiziert. #TIB wird nicht veraendert.
Der String kann sich oberhalb von HERE befinden.

] (--) 83,I "right-bracket"
(--) compiling
Schaltet den kompilierenden Zustand ein. Der Text vom Quelltext wird sukzessive kompiliert. Typische Benutzung s. LITERAL

\ (--) I "skip-line"
(--) compiling
Ignoriere den auf dieses Wort folgenden Text bis zum Ende der Zeile. s. C/L

\\ (--) I "skip-screen"
(--) compiling
Ignoriere den auf dieses Wort folgenden Text bis zum Ende des Blockes. s. B/BLK

\needs (--) "skip-needs"
Wird in der folgenden Art benutzt:
\needs <name>
Wird <name> in der Suchreihenfolge gefunden, so wird der auf <name> folgende Text bis zum Ende der Zeile ignoriert. Wird <name> nicht gefunden, so wird die Interpretation hinter <name> fortgesetzt.
Beispiel:\needs Editor 1+ load
Laedt den folgenden Block, falls EDITOR im Dictionary nicht vorhanden ist.

Fehlerbehandlung

(error (string --) "paren-error"
Dieses Wort steht normalerweise in der Variablen ERRORHANDLER und wird daher bei ABORT" und ERROR" ausgefuehrt. string ist dann die Adresse des auf ABORT" bzw. ERROR" folgenden Strings. (ERROR gibt das letzte Wort des Quelltextes gefolgt von dem String auf dem Bildschirm aus. Die Position des letzten Wortes im Quelltext, bei dem der Fehler auftrat, wird in SCR und R# abgelegt.

?pairs (n1 n2 --) "question-pairs"
Ist n1 <> n2 , so wird die Fehlermeldung "unstructured" ausgegeben. Dieses Wort wird benutzt, um die korrekte Schachtelung der Kontrollstrukturen zu ueberpruefen.

?stack (--) "question-stack"
Prueft, ob der Stack ueber- oder leerlaeuft. Der Stack laeuft leer, falls der Stackpointer auf eine Adresse oberhalb von S0 zeigt. In diesem Fall wird die Fehlermeldung "stack empty" ausgegeben. Der Stack laeuft ueber, falls der Stackpointer zwischen HERE und HERE + $100 liegt. In diesem Fall wird die Fehlermeldung "tight stack" ausgegeben, falls mehr als 31 Werte auf dem Stack liegen. Ist das nicht der Fall, so versucht das System, das zuletzt definierte Wort zu vergessen und es wird die Fehlermeldung "dictionary full" ausgegeben.

abort (--) 83,I
Leert den Stack, fuehrt END-TRACE STANDARDI/O und QUIT aus.

Abort" (flag) 83,I,C "abort-quote"
(--) compiling
Wird in der folgenden Form benutzt:
flag Abort" ccc"
Wird ABORT" spaeter ausgefuehrt, so geschieht nichts, wenn flag falsch ist. Ist flag wahr, so wird der Stack geleert und der Inhalt von ERRORHANDLER ausgefuehrt. Beachten Sie bitte, dass im Gegensatz zu ABORT kein END-TRACE ausgefuehrt wird.

diskerr (--)
Ein deferred Wort, das normalerweise mit (DISKERR vorbesetzt ist. DISKERR wird aufgerufen, wenn ein Fehler beim Massenspeicherzugriff auftrat. (DISKERR gibt dann die Meldung "error ! r to retry" aus. Wird anschliessend r gedrueckt, so wiederholt das System den Massenspeicherzugriff, der zum Fehler fuehrte. Wird eine andere Taste gedrueckt, so wird der Zugriff abgebrochen und die Meldung "aborted" ausgegeben. In diesem Fall wird die interne Verteilung der Blockpuffer nicht geaendert.

Error" (flag) I,C "error-quote"
(--) compiling
Dieses Wort entspricht ABORT" , jedoch mit dem Unterschied, dass der Stack nicht geleert wird.

errorhandler (-- adr)
adr ist die Adresse einer Uservariablen, deren Inhalt die Kompilationsadresse eines Wortes ist. Dieses Wort wird ausgefuehrt, wenn das flag, das ABORT" bzw. ERROR" verbrauchen, wahr ist. Der Inhalt von ERRORHANDLER ist normalerweise (ERROR.

warning (-- adr)
adr ist die Adresse einer Variablen. Ist der Inhalt der Variablen null, so wird die Warnung "exists" ausgegeben, wenn ein Wort redefiniert wird. Ist der Wert nicht null, so wird die Warnung unterdrueckt.

Sonstiges

'abort (--) "tick-abort"
Dies ist ein deferred Wort, das mit NOOP vorbesetzt ist. Es wird in ABORT ausgefuehrt, bevor QUIT aufgerufen wird. Es kann benutzt werden, um "automatisch" selbst definierte Stacks zu loeschen.

'cold (--) "tick-cold"
Dies ist ein deferred Wort, das mit NOOP vorbesetzt ist. Es wird in COLD aufgerufen, bevor die Einschaltmeldung ausgegeben wird. Es wird benutzt, um Geraete zu initialisieren oder Anwenderprogramme automatisch zu starten.

'quit (--) "tick-quit"
Dies ist ein deferred Wort, das normalerweise mit (QUIT besetzt ist. Es wird in QUIT aufgerufen, nachdem der Returnstack enleert und der interpretierende Zustand eingeschaltet wurde. Es wird benutzt, um spezielle Kommandointerpreter (wie z.B. im Tracer) aufzubauen.

'restart (--) "tick-restart"
Dies ist ein deferred Wort, das mit NOOP vorbesetzt ist. Es wird in RESTART aufgerufen, nachdem 'QUIT mit (QUIT besetzt wurde. Es wird benutzt, um Geraete nach einem Warmstart zu reinitialisieren.

(quit (--) "paren-quit"
Dieses Wort ist normalerweise der Inhalt von 'QUIT. Es wird von QUIT benutzt. Es akzeptiert eine Zeile von der aktuellen Eingabeeinheit, fuerht sie aus und druckt "ok" bzw. "compiling".

.status (--) "dot-status"
Dieses Wort ist ein deferred Wort, das vor dem Einlesen einer Zeile bzw. dem Laden eines Blocks ausgefuehrt wird. Es ist mit NOOP vorbesetzt und kann dazu benutzt werden, Informationen ueber den Systemzustand oder den Quelltext auszugeben.

bye (--)
Dieses Wort fuehrt FLUSH und EMPTY aus. Anschliessend wird der Monitor des Rechners angesprungen oder eine andere implementationsabhaengige Funktion ausgefuehrt.

cold (--)
Bewirkt den Kaltstart des Systems. Dabei werden alle nach der letzten Ausfuehrung von SAVE definierten Worte entfernt, die Uservariablen auf den Wert gesetzt, den sie bei SAVE hatten, die Blockpuffer neu initialisiert, der Bildschirm geloescht und die Einschaltmeldung "ultraFORTH-83 rev..." ausgegeben. Anschliessend wird RESTART ausgefuehrt.

end-trace (--)
Schaltet den Tracer ab, der durch "patchen" der Next-Routine arbeitet. Die Ausfuehrung des aufrufenden Wortes wird fortgesetzt.

noop (--)
Tut gar nichts.

R# (-- adr) "r-sharp"
adr ist die Adresse einer Variablen, die den Abstand des gerade editierten Zeichens vom Anfang des gerade editierten Screens enthaelt. Vergleiche (ERROR und SCR.

restart (--)
Bewirkt den Warmstart des Systems. Dieses Wort wird beim C64 durch Druecken von RUN/STOP - RESTORE aufgerufen. Es setzt 'QUIT , ERRORHANDLER und 'ABORT auf ihre normalen Werte und fuehrt ABORT aus.

scr (-- adr) 83 "s-c-r"
adr ist die Adresse einer Variablen, die die Nummer des gerade editierten Screens enthaelt. Vergleiche R# (ERROR und LIST .

Massenspeicher

>drive (block #drv -- block') "to-drive"
block' ist die Nummer des Blocks block auf dem Laufwerk #drv, bezogen auf das aktuelle Laufwerk (Vergleiche OFFSET und DRIVE) . Beispiel:
5 20 >drive block
holt den Block mit der Nummer 20 vom Laufwerk 5, egal welches Laufwerk gerade das aktuelle ist.

all-buffers (--)
Belegt den gesamten Speicherbereich zwischen dem Ende des Returnstacks und LIMIT mit Blockpuffern.

allotbuffer (--)
Fuegt der Liste der Blockpuffer noch einen weiteren hinzu, falls oberhalb vom Ende des Retunstacks dafuer noch Platz ist. FIRST wird entsprechend geaendert. Vergleiche ALL-BUFFERS

b/blk (-- &1024) "bytes pro block"
&1024 ist die Anzahl der Bytes in einem Block.

b/buf (-- n) "bytes pro buffer"
n ist die Laenge eines Blockpuffers incl. der Verwaltungsinformationen des Systems.

blk/drv (-- n) "blocks pro drive"
n ist die Anzahl der auf dem aktuellen Laufwerk verfuegbaren Bloecke.

block (u -- addr) 83
addr ist die Adresse des ersten Bytes des Blocks u in dessen Blockpuffer. Der Block u stammt aus dem File in FILE . Falls der Block in diesem Puffer UPDATEd und nicht der Block u ist, dann wird er auf den Massenspeicher zurueckuebertragen, bevor der Blockpuffer an den Block u vergeben wird. Befindet sich der Block u nicht in einem Blockpuffer, so wird er vom Massenspeicher in einen an ihn vergebenen Blockpuffer (s.o.) geladen. Eine Fehlerbedingung liegt vor, falls u keine Nummer fuer einen vorhandenen Block ist. Nur Daten im letzten Blockpuffer, der von BLOCK oder BUFFER vergeben wurde, sind gueltig. Alle anderen Blockpuffer duerfen nicht mehr als gueltig angenommen werden. Der Inhalt eines Blockpuffers sollte nur veraendert werden, wenn die Aenderungen auch auf den Massenspeicher uebertragen werden sollen.

buffer (u -- adr) 83
Vergibt einen Blockpuffer an den Block u. addr ist die Adresse des ersten Bytes des Blocks in seinem Puffer. Dieses Wort entspricht BLOCK , jedoch mit der Ausnahme, das, wenn der Block noch nicht im Speicher ist, er nicht vom Massenspeicher geholt wird. Daher ist der Inhalt dieses Blockpuffers nicht festgelegt.

convey (blk1 blk2 to.blk --)
Die Bloecke im Ursprungsbereich von blk1 bis blk2 inclusive werden in den Zielbereich ab Block to.blk verschoben. Die Bereiche duerfen sich ueberlappen. Ist blk2 kleiner als blk1, so wird "nein" ausgegeben und die Ausfuehrung abgebrochen.

copy (u1 u2 --)
Der Block u1 wird in den Block u2 kopiert. Der alte Inhalt des Blocks u2 ist verloren.

core? (blk file -- addr/false)"core-question"
Sofern sich der Block blk aus dem File file im Speicher befindet, ist addr die Adresse des ersten Bytes des Blocks in seinem Blockpuffer. Befindet sich der Block nicht im Speicher, so wird false als Ergebnis geliefert. Vergleiche BUFFER und FILE

drive (#drv --)
Selektiert das Laufwerk #drv als aktuelles. Dann liefert 0 BLOCK die Adresse des ersten Blocks auf diesem Laufwerk. Vergleiche >DRIVE und OFFSET.

drv? (block -- #drv) "drive-question"
#drv ist das Laufwerk, auf dem sich der Block block befindet. Vergleiche OFFSET >DRIVE und DRIVE.

empty-buffers (--)
Loescht den Inhalt aller Blockpuffer. UPDATEd Blockpuffer werden nicht auf den Massenspeicher zurueckgeschrieben.

file (-- addr)
addr ist die Adresse einer Uservariablen, deren Wert die Nummer des aktuellen Files, auf das sich alle Massenspeicheroperationen beziehen, ist. Typisch entspricht die Nummer eines Files der Adresse seines File Control Blocks. Ist der Wert von FILE Null, so wird direkt, ohne ein File, auf den Massenspeicher (z.B. die Sektoren einer Diskette) zugegriffen.

first (-- addr)
addr ist die Adresse einer Variablen, in der sich ein Zeiger auf den Blockpuffer mit der niedrigsten Adresse befindet. Vergleiche ALLOTBUFFER

flush (--) 83
Fuehrt SAVE-BUFFERS aus und loescht anschliessend alle Blockpuffer. Vergleiche EMPTY-BUFFERS

freebuffer (--)
Der Blockpuffer, auf den FIRST zeigt, wird, falls UPDATEd, auf den Massenspeicher gebracht und von der Liste der Blockpuffer entfernt. FIRST wird entsprechend veraendert. Der Speicherbereich, in dem sich dieser Puffer befand, steht damit zur Verfuegung. Gibt es nur noch einen Blockpuffer, so geschieht nichts.

limit (-- addr)
Unterhalb von addr befinden sich die Blockpuffer. Das letzte Byte des obersten Blockpuffers befindet sich in addr-1. Vergleiche ALL-BUFFERS ALLOTBUFFER

offset (-- addr)
addr ist die Adresse einer Uservariablen, die einen Offset enthaelt. Dieser Offset wird zu der Blocknummer addiert, die sich bei Aufruf von BLOCK BUFFER usw. auf dem Stack befindet. Die Summe ergibt die Nummer des Blocks, der vom Massenspeicher geholt wird.

prev (-- addr)
addr ist die Adresse einer Variablen, deren Wert der Anfang der Liste aller Blockpuffer ist. Der erste Blockpuffer in der Liste ist der zuletzt durch BLOCK oder BUFFER vergebene.

r/w (addr block file n -- flag)"r-w"
Ein deferred Wort, bei dessen Aufruf das systemabhaengige Wort ausgefuehrt wird, das einen Block vom Massenspeicher holt. Dabei ist addr die Anfangsadresse des Speicherbereichs fuer den Block block, file die Filenummer des Files, in dem sich der Block befindet und n=0, falls der Block vom Speicherbereich in den Massenspeicher gebracht werden soll. Ist n=1, so soll der Block vom Massenspeicher in den Speicherbereich gelesen werden. Das flag ist Falsch, falls kein Fehler auftrat, sonst Wahr.

save-buffers (--) 83
Der Inhalt aller Blockpuffer, die als UPDATEd gekennzeichnet sind, wird in ihre korrespondierenden Massenspeicherbloecke zurueckgeschrieben. Alle Blockpuffer werden als unveraendert gekennzeichnet, bleiben aber an ihre Bloecke vergeben.

update (--) 83
Der zuletzt mit BLOCK BUFFER usw. zugegriffene Block wird als veraendert gekennzeichnet. Bloecke mit solcher Kennzeichnung werden auf den Massenspeicher zurueckgeschrieben, wenn ihr Blockpuffer fuer einen anderen Block benoetigt oder wenn SAVE-BUFFERS ausgefuehrt wird. Vergleiche PREV

C64-spezifische Worte

(bload (-- flag)
laedt ein File von einem externen Geraet. Ein Fehler beim Laden wird durch flag <> FALSE angezeigt. Die File- Parameter muessen in der Zeropage abgelegt sein. Siehe (BSAVE .

(bsave (-- flag)
speichert ein File auf ein externes Geraet. Ist flag = FALSE , so war das Abspeichern erfolgreich, sonst ist flag <> FALSE. (BLOAD setzt voraus, dass die File- Parameter in der Zeropage abgelegt sind:

ADR	Typ	Bedeutung
0AE	Wort	Endadresse des Files +1
0B7	Byte	Laenge des Filenamens
0BA	Byte	Device Nummer
0BB	Wort	Adresse des Filenamens
0C1	Wort	Anfangsadresse des Files

Vergleiche (BLOAD .

1541r/w (adr blk file r/wf -- flag) "15-41-r-w"
liest oder schreibt einen Block Daten ($400 (&1024) Bytes) vom Diskettenblock blk nach adr bzw. von adr auf den Diskettenblock blk. Ist r/wf = FALSE , so wird auf die Diskette geschrieben, ist r/wf <> FALSE , so wird von der Diskette gelesen. 1541R/W ermittelt die zu lesenden bzw. zu beschreibenden Sektoren, fuehrt das Lesen bzw. Schreiben aus und prueft auf alle moeglichen Fehler. Vergleiche R/W .

file ist die Nummer des Files, dem der zu uebertragende Block zugeordnet ist. Zur Zeit ist ausschliesslich der Direktzugriff auf die Diskette realisiert. Deshalb wird eine Fehlerbehandlung eingeleitet, wenn file <> 0 ist.

?device (dev# --) "question-device"
prueft, ob das Geraet mit der Nummer dev# am seriellen Bus anwesend ist. Meldet sich das Geraet nicht, so wird eine Fehlerbehandlung eingeleitet. Ein Fehler liegt vor, wenn dev# nicht die Bedingung 4 =< dev# =< $0F (&15) erfuellt. Typische dev# sind 8 fuer das Diskettenlaufwerk und 4 fuer den Drucker.

bus! (8b --) "bus-store"
gibt 8b ueber den seriellen Bus aus. Ein Fehler liegt vor, wenn die Ausgabe nicht vorbereitet wurde. (Z.B. durch BUSOUT oder BUSOPEN).

bus@ (-- 8b) "bus-fetch"
holt 8b vom seriellen Bus. Ein Fehler liegt vor, wenn die Eingabe nicht vorbereitet wurde. (Z.B. durch BUSIN).

busclose (dev# 2nd --)
beendet die Ausgabe an oder die Eingabe vom Geraet mit der Nummer dev# in bzw. aus einer mit BUSOPEN eroeffnete Datei. BUSCLOSE braucht nicht mit BUSOFF abgeschlossen werden. Weiteres Verhalten siehe BUSOUT .

busin (dev# 2nd --)
bereitet die Eingabe ueber den seriellen Bus vom Geraet mit der Nummer dev# vor. Verhaelt sich sonst wie BUSOUT , siehe dort.

businput (adr u --)
holt u Zeichen vom seriellen Bus und legt sie im Speicher ab adr ab. Ein PAUSE wird ausgefuehrt. Ein Fehler liegt vor, wenn die Eingabe nicht vorbereitet wurde. (Z.B. durch BUSIN).

busoff (--)
gibt den seriellen Bus und den Semaphor I/O frei. Vergleiche BUSIN , BUSOUT , BUSOPEN , BUSCLOSE und I/O , LOCK , UNLOCK .

busopen (dev# 2nd --)
bereitet die Ausgabe ueber den seriellen Bus auf das Geraet mit der Nummer dev# vor und teilt dem Geraet mit, dass alle nachfolgenden Zeichen bis zum naechsten BUSOFF als Dateiname aufzufassen sind. Dient zur Vorbereitung von Ein- oder Ausgaben von bzw. in Dateien auf Diskette. Weiteres Verhalten siehe BUSOUT .

busout (dev# 2nd --)
bereitet die Ausgabe ueber den seriellen Bus auf das Geraet mit der Nummer dev# vor. Meldet sich das Geraet nicht, so wird eine Fehlerbehandlung eingeleitet, sonst bekommt das Geraet den Wert 2nd zugesandt. Zulaessige Werte sind 4 =< dev# =< $0F (&15) und 0 =< 2nd =< $1F (&31) anderenfalls liegt ein Fehler vor. Der Semaphor I/O wird gesperrt und damit der serielle Bus vor dem Zugriff durch andere Tasks geschuetzt (Vergleiche LOCK , I/O). Vergleiche BUSOFF , BUSIN , BUSOPEN , BUSCLOSE .

bustype (adr u --)
gibt u Zeichen, die ab adr im Speicher stehen, ueber den seriellen Bus aus. Ein PAUSE wird ausgefuehrt. Ein Fehler liegt vor, wenn die Ausgabe nicht vorbereitet wurde. (Z.B. durch BUSOUT oder BUSOPEN).

c64at (row col --) "c-64-at"
positioniert den Cursor in die Zeile row und die Spalte col. Ein Fehler liegt vor, wenn row > $18 (&24) oder col > $27 (&39) ist. Vergleiche AT .

c64at? (-- row col) "c-64-at-question"
ermittelt die aktuelle Cursorposition und legt die Nummer der Zeile row und die Nummer der Spalte col nacheinander auf den Stack. Vergleiche AT? .

c64cr (--) "c-64-c-r"
ein Wagenruecklauf wird auf die Console gegeben. Vergleiche CR . Es wird immer die Console angesprochen. Ein PAUSE wird ausgefuehrt.

c64decode (adr len0 key -- adr len1) "c-64-decode"
wertet key aus. Ist key weder #BS noch #CR , so wird key in der Speicherstelle adr + len0 abgelegt, das Zeichen als Echo zum Ausgabegeraet gesandt und len0 inkrementiert. Im Falle von #BS wird das letzte Echo geloescht und len0 dekrementiert, bei #CR wird len0 nicht veraendert und in die Variable SPAN kopiert. Vergleiche DECODE . Sollen andere Zeichen (z. B. Cursortasten) ausgewertet werden, so ist ein USERdecode zu schreiben und in die Input-Struktur einzuhaengen. Vergleiche INPUT: und EDIDECODE .

c64del (--) "c-64-del"
ueberschreibt das Zeichen links vom Cursor mit einem SPACE und positioniert den Cursor darauf. Vergleiche DEL . Siehe C64CR fuer weiteres Verhalten.

c64emit (8b --) "c-64-emit"
gibt 8b auf die Console aus. Alle nicht druckbaren (Steuer-) Zeichen werden durch einen Punkt ersetzt. Es wird immer die Console angesprochen. Ein PAUSE wird ausgefuehrt. Vergleiche EMIT und OUTPUT: .

c64expect (adr len --) "c-64-expect"
erwartet len Zeichen vom Eingabegeraet, die ab adr im Speicher abgelegt werden. Ein Echo der Zeichen wird ausgegeben. CR beendet die Eingabe vorzeitig. Ein abschliessendes Leerzeichen wird immer ausgegeben. Die Laenge der empfangenen Zeichenkette wird in der Variablen SPAN uebergeben. Vergleiche EXPECT . Siehe auch EDIEXPECT .

c64init (--) "c-64-init"
initialisiert den C64. Neben der normalen Reset-Initialisierung wird das basic-ROM abgeschaltet, Rahmen-, Bildschirm- und Zeichenfarbe aus INK-POT gesetzt, Repeat fuer alle Tasten eingeschaltet und der Modus Gross/Kleinschrift gewaehlt. C64INIT wird typisch durch COLD oder RESTART aufgerufen.

c64key (-- 8b) "c-64-key"
wartet auf einen Tastendruck. Waehrend der Wartezeit wird PAUSE ausgefuehrt. Der cbm-Code des Tastendrucks wird auf den Stack gelegt. Steuerzeichen werden nicht ausgewertet, sondern unveraendert abgeliefert. Vergleiche KEY .

c64key? (-- f) "c-64-key-question"
prueft, ob eine Taste gedrueckt wurde und hinterlaesst dann f = TRUE , wurde keine Taste gedrueckt, so ist f = FALSE . Vergleiche KEY? .

c64page (--) "c-64-page"
loescht den Bildschirm und positioniert den Cursor in die linke obere Ecke. Vergleiche PAGE . Siehe C64CR fuer weiteres Verhalten.

c64type (adr len --) "c-64-type"
gibt den String, der im Speicher bei adr beginnt und die Laenge len hat, auf die Console aus. Wirkt immer auf die Console. Alle nicht druckbaren (Steuer-) Zeichen werden durch einen Punkt ersetzt. Ein PAUSE wird ausgefuehrt. Vergleiche TYPE , OUTPUT: und C64EMIT .

con! (8b --) "con-store"
gibt 8b auf die CONsole aus. Es wird immer der Bildschirm angesprochen. Alle Steuerzeichen werden zur Console gesandt. Ein PAUSE wird ausgefuehrt.

curoff (--)
schaltet den Cursor aus.

curon (--)
schaltet den Cursor ein.

derror? (-- f) "derror-question"
prueft den Fehlerkanal des aktuellen Diskettenlaufwerks. Ist die Fehlernummer kleiner als $0A (&10) (das heisst, kein Fehler), so ist f = FALSE . Liegt ein Fehler vor, so wird f = TRUE und die ganze Fehlermeldung in der Form NN,Fehlertext,TT,SS ausgegeben. Dabei ist NN die Fehler-Nummer, TT die Track- uns SS die Sektor-Nummer, bei der der Fehler auftrat (alle Zahlen in decimal).

diskclose (--) "disk-close"
schliesst den mit DISKOPEN eroeffneten Disketten-kanal wieder. DISKCLOSE ist zum Abschluss von DISKOPEN vorgesehen. Vergleiche READSECTOR und WRITESECTOR .

diskopen (-- f) "disk-open"
oeffnet den Disketten-Kanal $0D (&13) fuer nachfolgende Lese- oder Schreib-Vorgaenge im direkten Zugriff auf die Diskette. DISKOPEN sollte die Aus- und Eingabe mit READSECTOR und WRITESECTOR vorbereiten. Ist das Oeffnen des Disketten-Kanals erfolgreich, so ist f = FALSE , sonst ist f = TRUE .

display (--)
ein mit OUTPUT: definiertes Wort, das den Bildschirm als Ausgabegeraet setzt, wenn es ausgefuehrt wird. Die Worte EMIT , CR , TYPE , DEL , PAGE , AT und AT? beziehen sich dann auf den Bildschirm.

findex (from to --)
liest, unter Umgehung des BLOCK-Mechanismus, die ersten Zeilen der Blocks from bis to einschliesslich nach PAD und gibt sie aus. Der Bereich PAD bis PAD $100 + wird von FINDEX benutzt. FINDEX kann mit einer beliebigen Taste angehalten und mit RUN/STOP abgebrochen werden. Vergleiche dazu STOP? . Die Block-Buffer werden nicht veraendert. Vergleiche auch READSECTOR .

getkey (-- 8b)
holt den cbm-Code des jeweils naechsten Tastendrucks und legt ihn auf den Stack. War keine Taste gedrueckt, so ist 8b = FALSE . (Vergleiche KEY?) GETKEY liest direkt aus dem Tastaturpuffer.

i/o (-- semaphoradr) "i-o"
ein Semaphor (Ampel) (eine spezielle Variable). I/O schuetzt den seriellen Bus vor dem Zugriff durch andere Prozesse, wenn er von einem benutzt wird. Ist der Inhalt von I/O FALSE , so ist der Weg zum seriellen Bus fuer alle Prozesse freigegeben. Alle im System enthaltenen Routinen sind abgesichert. Der Benutzer muss bei eigenen Routinen selbst fuer eine Absicherung sorgen. Vergleiche LOCK , UNLOCK und die Beschreibung des Taskers.

index (from to --)
liest die BLOCKs from bis to einschliesslich und gibt deren erste Zeilen aus. INDEX kann mit einer beliebigen Taste angehalten und mit RUN/STOP abgebrochen werden. Vergleiche dazu STOP? . Die ersten Zeilen von Screens enthalten typisch Kommentare, die den Inhalt charakterisieren.

ink-pot (-- adr)
ein Byte-Feld von 3 mal 4 Byte Laenge, in denen fuer 3 verschiedene Zwecke (Einschalt-, Editor- und User-Farben) die Rahmen-, Bildschirm-, Zeichen-Farbe und ein Dummy-Byte abgelegt sind. Zur Farbaenderung ist die gewuenschte Farb-Nummer (0-&15) in das entsprechende Byte zu schreiben.

keyboard (--)
ein mit INPUT: definiertes Wort, das als Eingabegeraet die Tastatur setzt. Die Worte KEY , KEY? , DECODE und EXPECT beziehen sich auf die Tastatur. Steuerzeichen werden nicht ausgewertet. Siehe C64KEY , C64KEY? , C64DECODE und C64EXPECT. Vergleiche INPUT: , STANDARDI/O und EDIBOARD .

printable? (8b -- 8b f) "printable"
f ist TRUE , wenn 8b ein druckbares Zeichen ist, sonst ist f = FALSE .

readsector (adr tra# sec# -- f)
liest alle $100 (&256) Bytes des Sektors sec# von Spur tra# der Diskette im aktuellen Laufwerk ueber den seriellen Bus und legt sie ab adr im Speicher ab. War das Lesen erfolgreich, so ist f = FALSE , sonst ist f <> FALSE und die Fehlermeldung des Laufwerks wird ausgegeben. Im Fehlerfall wird der Speicherinhalt ab adr nicht veraendert. Der Disketten-Kanal $0D (&13) muss fuer READSECTOR offen sein (vergleiche DISKOPEN).

writesector (adr tra# sec# -- f)
schreibt $100 (&256) Bytes, die ab adr im Speicher stehen, ueber den seriellen Bus in den Sektor sec# der Spur tra# auf der Diskette im aktuellen Laufwerk. War das Schreiben erfolgreich, so ist f = FALSE , sonst ist f <> FALSE und die Fehlermeldung des Laufwerks wird ausgegeben. Der Disketten-Kanal $0D (&13) muss fuer WRITESECTOR offen sein (vergleiche DISKOPEN).

Multitasking Worte

's (Tadr -- usradr) "tick-s"
wird benutzt in der Form:
... <taskname> 's <uname> ...
liest den Namen einer Uservariablen <uname> und hinterlaesst die Adresse usradr dieser Uservariable in der durch Tadr gekennzeichneten Task. Typisch wird Tadr durch Nennung von <taskname> erzeugt. Eine Fehlerbedingung liegt vor, wenn <uname> nicht der Name einer Uservariablen ist. Vergleiche USER und TASK . 'S ist fuer die Veraenderung des Inhalts von User-Variablen einer Task durch eine andere Task vorgesehen.

activate (Tadr --)
aktiviert die Task, die durch Tadr gekennzeichnet ist, und weckt sie auf. Tadr wird typisch durch Nennung eines Task-Namens erzeugt. Vergleiche SLEEP , STOP , PASS , PAUSE , UP@ , UP! und WAKE .

lock (semadr --)
der Semaphor, dessen Adresse auf dem Stack liegt, wird von der Task, die LOCK ausfuehrt, blockiert. Dazu prueft LOCK den Inhalt von semadr. Zeigt der Inhalt an, dass eine andere Task den Semaphor blockiert hat, so wird PAUSE ausgefuehrt, bis der Semaphor freigegeben ist. Ist der Semaphor freigegeben, so schreibt LOCK ein Kennzeichen der Task, die LOCK ausfuehrt, in den Semaphor und sperrt ihn damit fuer alle anderen Tasks. Den Code zwischen semadr LOCK ... und ... semadr UNLOCK kann also immer nur eine Task ausfuehren. Vergleiche UNLOCK und die Beschreibung des Taskers.

multitask (--)
schaltet das Multitasking ein. Das Wort PAUSE ist nach Ausfuehrung von MULTITASK keine NOOP-Funktion mehr, sondern gibt die Kontrolle ueber den Prozessor an eine andere Task weiter.

pass (n0 ... nr-1 Tadr r --)
aktiviert die Task, die durch Tadr gekennzeichnet ist, und weckt sie auf. Tadr wird typisch durch Nennung eines Task-Namens erzeugt. r gibt die Anzahl der Parameter n0 bis nr-1 an, die vom Stack der PASS ausfuehrenden Task auf den Stack der durch Tadr gekennzeichneten Task uebergeben werden. Die Parameter n0 bis nr-1 stehen der durch Tadr gekennzeichneten Task zur weiteren Verarbeitung zur Verfuegung. Vergleiche STOP , ACTIVATE , PAUSE , UP@ und UP! .

pause (--)
eine NOOP-Funktion, wenn der Singletask-Betrieb eingeschaltet ist; bewirkt jedoch, bei aktiviertem Multitasking, dass die Task, die PAUSE ausfuehrt, die Kontrolle ueber den Prozessor an eine andere Task abgibt. Existiert nur eine Task, oder schlafen alle anderen Tasks, so wird die Kontrolle unverzueglich an die Task zurueckgegeben, die PAUSE ausfuehrte. Ist mindestens eine andere Task aktiv, so wird die Kontrolle des Prozessors von dieser uebernommen und erst bei erneuter Ausfuehrung von PAUSE oder STOP an eine andere Task weitergegeben. Da die Tasks zyklisch miteinander verkettet sind, erhaelt die Task, die zuerst PAUSE ausfuehrte, irgendwann die Kontrolle zurueck. Eine Fehlerbedingung liegt vor, wenn eine Task weder PAUSE noch STOP ausfuehrt. Vergleiche STOP , MULTITASK und SINGLETASK .

rendezvous (semadr --)
gibt den Semaphor mit der Adresse semadr frei und fuehrt PAUSE aus, um anderen Tasks den Zugriff auf das durch diesen Semaphor geschuetzte Geraet zu ermoeglichen. Anschliessend wird LOCK ausgefuehrt, um das Geraet zurueck zu erhalten. Vergleiche LOCK und UNLOCK .

singletask (--)
schaltet das Multitasking aus. Das Wort PAUSE ist nach Ausfuehrung von SINGLETASK eine NOOP-Funktion. Eine Fehlerbedingung liegt vor, wenn eine Hintergrundtask SINGLETASK ohne anschliessendes MULTITASK ausfuehrt, da die MAIN- oder TERMINAL-TASK dann nicht mehr die Kontrolle ueber den Prozessor bekommt. Vergleiche UP@ und UP! .

sleep (Tadr --)
bringt die Task, die durch Tadr gekennzeichnet ist, zum Schlafen. Tadr wird typisch durch Nennung eines Task-Namens erzeugt. SLEEP hat den gleichen Effekt, wie die Ausfuehrung von STOP durch die Task selbst. Der Unterschied ist, dass STOP in der Regel ein Endpunkt in der Bearbeitung ist, SLEEP trifft die Task zu einem nicht vorhersehbaren Zeitpunkt, so dass die laufende Arbeit der Task unterbrochen wird. Vergleiche WAKE .

stop (--)
bewirkt, dass die Task, die STOP ausfuehrt, sich schlafen legt. Der Inhalt des IP (Interpretive Pointer), des RP (Returnstack Pointer) und des SP (Stack Pointer) werden gesichert, dann wird die Kontrolle ueber den Prozessor an die naechste Task abgegeben. Diese Aktionen werden ebenfalls von PAUSE ausgefuehrt (siehe dort), der Unterschied zu

PAUSE ist, dass die Task bei STOP inaktiv hinterlassen wird, bei PAUSE dagegen aktiv. Vergleiche auch ACTIVATE , PASS , WAKE , SLEEP , UP@ und UP! .

Task (rlen slen --)
wird benutzt in der Form:
rlen slen Task <cccc>
Task ist ein definierendes Wort, das eine Task - den Arbeitsbereich fuer ein weiteres Programm, das gleichzeitig zu den schon laufenden Programmen ablaufen soll - einrichtet. Die Task erhaelt den Namen cccc, hat einen Stack von der Groesse slen und einen Returnstack von rlen Bytes. Im Stack-Bereich liegen das task-eigene Dictionary (einschliesslich PAD), das in Richtung zu hoeheren Adressen waechst und der Stack, der zu niedrigeren Adressen hin waechst. Im Returnstack-Bereich befinden sich die task-eigene Userarea (waechst zu hoeheren Adressen), und der Returnstack (wird gegen kleinere Adressen groesser). Eine Task ist damit eine verkleinertes Abbild des gesamten ultraFORTH-Systems.

Die Ausfuehrung von cccc in einer beliebigen Task hinterlaesst die gleiche Adresse, die die Task cccc selbst mit UP@ erzeugt. Diese Adresse wird von 'S , ACTIVATE , LOCK , PASS , SLEEP , TASK und WAKE benutzt.

tasks (--)
listet die Namen aller eingerichteten Tasks und zeigt, ob sie schlafen oder aktiv sind.

unlock (semadr --)
gibt den Semaphor, dessen Adresse auf dem Stack liegt, fuer alle Tasks frei. Ist der Semaphor im Besitz einer anderen Task, so muss die UNLOCK ausfuehrende Task mit PAUSE auf die Freigabe warten. Vergleiche LOCK und die Beschreibung des Taskers.

up@ (-- Tadr) "u-p-fetch"
liefert die Adresse Tadr des ersten Bytes der Userarea der Task, die UP@ ausfuehrt. Tadr ist die Adresse, die jede Task kennzeichnet. Vergleiche dazu 'S , ACTIVATE , LOCK , PASS , SLEEP , TASK und WAKE . In der Userarea sind Variablen und andere Datenstrukturen hinterlegt, die jede Task fuer sich haben muss. Vergleiche UP! .

up! (adr --) "u-p-store"
richtet den UP (User Pointer) auf adr. Vergleiche UP@ .

wake (Tadr --)
weckt die Task, die durch Tadr gekennzeichnet ist, auf. Tadr wird typisch durch Nennung eines Task-Namens erzeugt. Die Task fuehrt ihren Code dort weiter aus, wo sie durch SLEEP angehalten wurde oder wo sie sich selbst durch STOP beendet hat (Vorsicht!) . Vergleiche SLEEP, STOP, ACTIVATE und PASS .

Input und Output Worte

#bs (-- n) "number-b-s"
n ist der Wert, den man durch KEY erhaelt, wenn die Backspace- (Delete-) Taste gedrueckt wird.

#cr (-- n) "number-c-r"
eine Konstante, die den Wert liefert, den man durch KEY erhaelt, wenn die Return-Taste gedrueckt wird.

#tib (-- adr) 83 "number-t-i-b"
eine Variable, die die Laenge des aktuellen Textes im Text-Eingabe-Puffer haelt. Vergleiche TIB .

-trailing (adr +n0-- adr +n1) 83 "dash-trailing"
adr ist die Anfangsadresse und +n0 die Laenge eines Strings. -TRAILING veraendert +n0 so zu +n1, dass eventuell abschliessende Leerzeichen nicht mehr in der neuen Stringlaenge +n1 enthalten sind. Der String selbst bleibt unangetastet. Ist +n0 = 0, so ist auch +n1 = 0. Besteht der ganze String aus Leerzeichen, so ist +n1 = 0.

. (n --) 83 "dot"
druckt n vorzeichenbehaftet aus.

." (--) 83 I C "dot-quote"
(--) compiling
wird in :-Definitionen in der Form verwendet:
: <name> ... ." cccc" ... ;
Der String cccc wird bis zum abschliessenden " so compiliert, dass bei Ausfuehrung von <name> der String cccc ausgedruckt wird. Das Leerzeichen nach ." und das abschliessende " sind Pflicht und gehoeren nicht zum String.

.((--) 83 I "dot-paren"
(--) compiling
wird in der Form:
... .(cccc) ...
benutzt und druckt den String cccc bis zur abschliessenden Klammer sofort aus. Das Leerzeichen nach .(und die schliessende Klammer sind Pflicht und gehoeren nicht zum String.

.r (n +n --) "dot-r"
druckt die Zahl n in einem +n langen Feld mit Vorzeichen rechtsbuendig aus. Reicht +n nicht zur Darstellung der Zahl aus, so wird ueber den rechten Rand hinaus ausgegeben. Die Zahl n wird in jedem Fall vollstaendig dargestellt.

>tib (-- adr) 83 "to-tib"
adr ist die Adresse eines Zeigers auf den Text-Eingabe-Puffer. Siehe TIB .

?cr (--) "question-c-r"
prueft, ob in der aktuellen Zeile mehr als C/L - $0A (&10) Zeichen ausgegeben wurden und fuehrt dann CR aus.

at (row col --)
positioniert die Schreibstelle eines Ausgabegeraetes in die Zeile row und die Spalte col. AT ist eines der ueber OUTPUT vektorisierten Worte.

at? (-- row col) "at-question"
ermittelt die aktuelle Position der Schreibstelle eines Ausgabe-Geraetes und legt Zeilen- und Spaltennummer auf den Stack. Eines der OUTPUT-Worte.

base (-- adr) 83 U
adr ist die Adresse einer Uservariablen, die die Zahlenbasis enthaelt, die zur Wandlung von Zahlen-Ein- und Ausgaben benutzt wird.

bl (-- 16b) "b-l"
16b ist der ASCII-Wert fuer das Leerzeichen.

c/l (-- +n) "characters-per-line"
+n ist die Anzahl der Zeichen pro Bildschirm-Zeile.

col (-- u)
u ist die Spalte in der die Schreibstelle eines Ausgabe-Geraetes sich gerade befindet. Vergleiche ROW und AT? .

cr (--) 83 "c-r"
bewirkt, dass die Schreibstelle eines Ausgabe-Geraetes an den Anfang der naechsten Zeile verlegt wird. Eines der OUTPUT-Worte.

d. (d --) "d-dot"
druckt d vorzeichenbehaftet aus.

d.r (d +n --) "d-dot-r"
druckt d vorzeichenbehaftet in einem +n Zeichen breiten Feld rechtsbuendig aus. Reicht +n nicht zur Darstellung der Zahl aus, so wird ueber den rechten Rand hinaus ausgegeben. Die Zahl d wird in jedem Fall vollstaendig dargestellt.

decimal (--)
stellt BASE auf $0A (&10) ein. Alle Zahlenein- und Ausgaben erfolgen im dezimalen Zahlensystem.

decode (adr +n0 key -- adr +n1)
wertet key aus. Typisch werden normale druckbare ASCII-Zeichen in die Speicherstelle adr + +n0 uebertragen, als Echo zum Ausgabegeraet gesandt und +n0 inkrementiert. Andere Zeichen (#BS, #CR, Steuercodes) koennen andere Aktionen zur Folge haben. Eines der ueber INPUT vektorisierten Worte. Vergleiche C64DECODE . Wird von EXPECT benutzt.

del (--)
loescht das letzte ausgesandte Zeichen. Eins der OUTPUT-Worte. Bei Druckern ist die korrekte Funktion nicht garantiert.

emit (16b --) 83
die unteren 7 Bit (commodore - Benutzer: Achtung: die unteren 8 Bit) werden ausgegeben. Ist das Zeichen nicht druckbar, (insbesondere alle Steuercodes) so wird stattdessen ein "." ausgegeben. Eines des OUTPUT-Worte.

expect (adr +n --) 83
empfaengt Zeichen und speichert sie im Speicher. Die Uebertragung beginnt bei adr und setzt sich zu hoeheren Adressen fort, bis ein Return erkannt oder +n Zeichen uebertragen sind. Ein Return wird nicht mit abgespeichert. Wenn +n = 0 ist, so werden keine Zeichen uebertragen. Alle empfangenen Zeichen werden als Echo, statt des Return wird ein Leerzeichen ausgegeben. Vergleiche SPAN . Eines der ueber INPUT vektorisierten Worte.

hex (--)
stellt BASE auf $10 (&16) ein. Alle Zahlenein- und Ausgaben erfolgen im hexadezimalen Zahlensystem.

input (-- adr) U
adr ist die Adresse einer Uservariablen, die einen Zeiger auf ein Feld von (zur Zeit) 4 Kompilations-Adressen enthaelt, die fuer ein Eingabe-Geraet die Funktionen KEY KEY? DECODE und EXPECT realisieren. Vergleiche die gesonderte Beschreibung der INPUT- und OUTPUT-Struktur.

key (-- 16b) 83
empfaengt ein Zeichen von einem Eingabe-Geraet. Die niederwertigen 7 Bit (commodore 8 Bit) enthalten den ASCII- (commodore-) Code des zuletzt empfangenen Zeichens. Alle gueltigen ASCII- (commodore-) Codes koennen empfangen werden. Steuerzeichen werden nicht ausgewertet, sondern so, wie sie

sind, abgeliefert. Es wird kein Echo ausgesandt. KEY wartet, bis tatsaechlich ein Zeichen empfangen wurde. Eines der INPUT-Worte.

key? (-- flag) "key-question"
flag ist TRUE, wenn ein Zeichen zur Eingabe bereitsteht, sonst ist flag FALSE . Eins der INPUT-Worte.

list (u --) 83
zeigt den Inhalt des Screens u. SCR wird auf u gesetzt. Siehe BLOCK.

l/s (-- +n) "lines-per-screen"
+n ist die Anzahl der Zeilen pro Bildschirmseite.

output (-- adr) U
adr ist die Adresse einer Uservariablen, die einen Zeiger auf ein Feld von (zur Zeit) 7 Kompilations-Adressen enthaelt, die fuer ein Ausgabe-Geraet die Funktionen EMIT , CR , TYPE , DEL , PAGE , AT und AT? realisieren. Vergleiche die gesonderte Beschreibung der INPUT- und OUTPUT-Struktur.

page (--)
bewirkt, dass die Schreibstelle eines Ausgabegeraetes auf eine leere neue Seite bewegt wird. Vergleiche C64PAGE. Eines der OUTPUT-Worte.

query (--) 83
Zeichen werden von einem Eingabe-Geraet geholt und in den Text-Eingabe-Puffer, der bei TIB beginnt, uebertragen. Die Uebertragung endet beim Emfang von Return oder wenn die Laenge des Text-Eingabe-Puffers erreicht ist. Die Werte von >IN und BLK werden auf 0 gesetzt und SPAN wird nach #TIB kopiert. Um Text aus dem Puffer zu lesen, kann WORD benutzt werden. Siehe EXPECT und "Quelltext".

row (-- n)
n ist die Zeile, in der die Schreibstelle eines Ausgabe-Geraetes sich gerade befindet. Vergleiche COL und AT? .

space (--) 83
sendet ein Leerzeichen an das Ausgabe-Geraet.

spaces (+n --) 83
sendet +n Leerzeichen an ein Ausgabe-Geraet. Ist +n = 0, so wird nichts ausgesandt.

span (-- adr) 83
der Inhalt der Variablen SPAN gibt an, wieviele Zeichen vom letzten EXPECT uebertragen wurden. Siehe EXPECT .

standardi/o (--) "standard-i-o"
stellt sicher, dass die beim letzten SAVE bestimmten Ein- und Ausgabegeraete wieder eingestellt sind.

stop? (-- flag) "stop-question"
steht vom Eingabe-Geraet ein Zeichen zur Verfuegung, so wird es geholt. Ist es #CR (commodore RUN/STOP bzw. Ctrl-C), so ist flag = TRUE, sonst wird auf das naechste Zeichen gewartet. Ist dieses jetzt = #CR (RUN/STOP) so wird STOP? mit TRUE verlassen, sonst mit FALSE .

tib (-- adr) 83 "tib"
liefert die Adresse des Text-Eingabe-Puffers. Er wird benutzt, um die Zeichen vom Quelltext des aktiven Eingabe-Geraetes zu halten. Er kann mindestens $50 (&80) Zeichen aufnehmen. Siehe >TIB .

type (adr +n --) 83
sendet +n Zeichen, die ab adr im Speicher abgelegt sind, an das aktive Ausgabegeraet. Ist +n = 0, so wird nichts ausgegeben.

u. (u --) "u-dot"
die Zahl u wird vorzeichenlos ausgedruckt.

u.r (u +n --) "u-dot-r"
druckt die Zahl u in einem +n langen Feld ohne Vorzeichen rechtsbuendig aus. Reicht +n nicht zur Darstellung der Zahl aus, so wird ueber den rechten Rand hinaus ausgegeben. Die Zahl u wird in jedem Fall vollstaendig dargestellt.

Ergänzungen / Berichtigungen des Glossars

Im folgenden werden Änderungen und Ergänzungen des Glossars zusammengefaßt. Hierbei wurden außer Worten des Forth-Kerns auch einige andere - im Lieferumfang enthaltene - Worte mitaufgenommen, um die Übersicht zu verbessern.
Vergleiche hierzu auch "Änderungen seit rev. 3.5"

Nachtrag

Create (--) 83

Ein definierendes Wort, das in der Form
Create <name>
benutzt wird. Es erzeugt einen Kopf für <name>. Die nächste freie Stelle im Dictionary (vergleiche **HERE** und **DP**) ist nach einem **CREATE** <name> das erste Byte des Parameterfeldes von <name>. Wenn <name> ausgeführt wird, legt es die Adresse seines Parameterfeldes auf den Stack. **CREATE** reserviert keinen Speicherplatz im Parameterfeld von <name>.
Das Verhalten von <name> kann mit **DOES>** verändert werden.

Create: (--)

Ein definierendes Wort, das in der Form
Create: <name> .. ;
benutzt wird. Es wirkt wie : mit der Ausnahme, daß bei Ausführung von <name> die Parameterfeldadresse von <name> auf den Stack gebracht, das Wort jedoch nicht ausgeführt wird.

exit (--) 83,C

Wird in einer :-Definition benutzt. Bei Ausführung von **EXIT** wird in das diese :-Definition aufrufende Wort zurückgekehrt. Eine Fehlerbedingung besteht, wenn das oberste Element des Returnstacks keine Rückkehradresse enthält. **EXIT** darf nicht innerhalb von **DO .. LOOP** verwendet werden.

order (--)

Bei Aufruf wird die aktuelle Suchreihenfolge (s. "Definition der Begriffe") ausgegeben. Anschließend wird das Vokabular ausgegeben, in das neue Worte eingetragen werden.

C64- und C16-spezifische Worte

(drv (--adr)

adr ist die Adresse einer Variablen, die das aktuelle Laufwerk enthält. Der Inhalt ist i.a. gleich 0. Wenn Diskettenzugriffe nicht über den Blockmechanismus, sondern z.B. über **READSECTOR** bzw. **WRITESECTOR** gehen, sollte (DRV gesetzt werden.

(64 (--) I
(16 (--) I

Diese Worte ermöglichen es maschinenspezifische Teile in einer gemeinsamen Quelle zu vereinigen. Typische Benutzung:

```
...WorteFürAlleMaschinen...
(64 ......WorteNurFürC64..... C)
(16 ......WorteNurFürC16..... C)
...WorteFürAlleMaschinen...
```

Beim C64 bewirkt (64 nichts, (16 ignoriert alles bis C) . Beim C16 bewirkt (16 nichts, (64 ignoriert alles bis C).

C) (--) I

Tut nichts und zwar sofort. Vgl. **NOOP** (16 (64.

c64fkeys (--)

Dieses Wort gibt es nur auf dem C16. Es stellt die Funktionstastenbelegung des C64 her.

Tools

cpush (adr u --)
Es werden analog **PUSH** u bytes ab inclusive adr gesichert. Beim nächsten **EXIT** oder **UNNEST** werden sie zurückgespeichert.

debug (--)
Benutzt in der Form:
debug <name>
Hierbei ist <name> ein ausführbares Wort. Aktiviert den Tracer, so daß das Wort <name> schrittweise ausgeführt wird.

endloop (--)
Deaktiviert den Tracer für das angezeigte Wort; er bleibt jedoch für alle folgenden Worte aktiv. Ist das angezeigte Wort **(LOOP** oder **(+LOOP** oder ein von **REPEAT** oder **UNTIL** kompilierter Sprung, so kann damit das (wiederholte) Tracen der Schleife unterdrückt werden.

nest (--)
Weist den Tracer an, das angezeigte Wort ebenfalls zu tracen.

trace' (--)
Benutzt in der Form:
Trace' <name>
Wirkt wie **DEBUG** <name> , zusätzlich wird jedoch das zu tracende Wort anschließend ausgeführt.

unnest (--)
Weist den Tracer an, das getracete Wort zuende auszuführen und erst ab dem aufrufenden Wort wieder zu tracen.

unbug (--)
Deaktiviert den Tracer. Die Ausführung wird fortgesetzt.

Kassettenversion

\IF (--) I
Benutzt in der Form
\IF <name> <worte> ..
Wenn <name> in der aktuellen Suchreihenfolge gefunden wird, werden die folgenden Worte bis zum Zeilenwechsel ausgeführt, sonst geschieht nichts. Gegenstück zu **\NEEDS**

(rd (--adr)
adr ist die Adresse einer Variablen, die die Anfangsadresse der aktuellen Ramdisk enthält bzw. 0 wenn explizit keine Ramdisk existiert. Zum Format der Ramdisk vgl. die Shadows des Quelltextes. Vgl. auch **RD**

.rd (--)
Druckt zentrale Informationen über die Ramdisk aus.

7>c (8b--7b) "seven-to-char
Zur Rück-Umwandlung von mit C>7 erzeugten 7-bit-Buchstaben in Buchstaben. Funktioniert nur für den von ultraFORTH benutzten Zeichensatz sicher.

autoload (--adr)
adr ist die Adresse einer Variablen. Ist sie ungleich 0, so wird beim nächsten **TAPEINIT** eine Ramdisk geladen. Wird i.a. nur vor **SAVESYSTEM** benutzt.

binary (u--u)
u ist die BlockNummer eines Blockes, der nicht komprimiert werden soll. Dies ist für binäre Ramdisk-Blöcke erforderlich, da sie sonst verändert würden. Der Block belegt ab Deklaration ca. 1024 Bytes.

bload (adr1 adr3 8b-- adr2)
Ab adr1 wird ein File mit dem 8b langen Namen, der ab adr3 im Speicher ahgelegt ist, vom mit device gesetzten Gerät gelesen. Diverse Fehlerbedingungen werden behandelt. adr2 ist die Endadresse des geladenen Files plus 1 . Beim C16/C64 hat der Benutzer bei allen Lade-Operationen Sorge zu tragen, daß genug Platz vorhanden ist, sonst folgen undefinierte Resultate. (System-Absturz)

bsave (adr1 adr2 adr3 8b--)
Der Bereich von adr1 bis exklusive adr2 wird mit dem 8b langen Namen, der ab adr3 im Speicher abgelegt ist, auf das mit device gesetzte Gerät geschrieben. Diverse Fehlerbedingungen werden behandelt.

c>7 (8b--7b) "char-to-seven"
Zur Umwandlung von Buchstaben in 7-bit-Buchstaben. Die Zurückverwandlung mit 7>C funktioniert nur für den von ultraFORTH benutzten Zeichensatz sicher.

cload (adr1 adr3 8b u1--adr2 u2)
Lädt ab adr1 das File mit dem 8b langen Namen, der bei adr3 steht von dem Gerät der Nummer u1. u1 enthält beim C16/C64 im unteren Byte die Geräte-Nummer und im oberen Byte die Sekundäradress (i.a.=0). adr2 ist die Endadresse+1 des geladenen Files. u2 ist eine Fehlerbeschreibung, die zum Aufruf von **DERR?** benutzt werden sollte. Beim C16/C64 hat der Benutzer bei allen Lade-Operationen Sorge zu tragen, daß genug Platz vorhanden ist. Sonst folgen undefinierte Resultate. (System-Absturz)

commodore (--)
Setzt den Kassettenrekorder im Commodore-Format als aktuelles Ausgabegerät. Vgl.: **DEVICE SUPERTAPE FLOPPY**

compress (adr1 adr2 u1--u2)
Zum Komprimieren von Forth-Quelltexten. Beginnend bei adr1 werden u1 Bytes zur adr2 komprimiert. Die Länge des komprimierten Bereichs ist u2 . Sie beträgt ca. 30-50% der ursprünglichen Länge. Vgl.: **EXPAND**

csave (adr1 adr2 adr3 8b u1--u2)
Sichert den Speicherbereich von adr1 bis exklusive adr2 unter dem 8b langen Namen, der bei adr3 steht auf das Gerät der Nummer u1. u1 enthält beim C16/C64 im unteren Byte die Geräte-Nummer und im oberen Byte die Sekundäradresse (i.a.=0). u2 ist eine Fehlerbeschreibung, die zum Aufruf von **DERR?** benutzt werden sollte.

derr? (u--flag)
Wird nach **CLOAD** und **CSAVE** benutzt. Falls ein Fehler aufgetreten ist, gibt es eine u entsprechende Fehlermeldung aus. flag ist true, wenn ein Fehler aufgetreten ist, sonst false.

device (--adr)
adr ist die Adresse einer Variablen, die im niederwertigen Byte die Geräte-Nummer des aktuellen Gerätes enthält und im höherwertigen Byte die Sekundäradresse (i.a.=0). Vgl.: COMMODORE SUPERTAPE FLOPPY

expand (adr1 adr2 u1--u2)
Zum Expandieren von komprimierten Forth-Quelltexten. Beginnend bei adr1 werden u1 Bytes zur adr2 expandiert. Die Länge des expandierten Bereichs wird in u2 zurückgegeben. Vgl.: **COMPRESS**

floppy (--)
Setzt das Diskettenlaufwerk als aktuelles Ausgabegerät. Vgl.: **DEVICE COMMODORE SUPERTAPE**

id" (--)
Benutzt in der Form
id" ccc"
Setzt den Namen der aktuellen Ramdisk auf RD.ccc

loadramdisk (--)
Es wird eine neue Ramdisk eingerichtet und vom aktuellen Gerät geladen. Beim C16/C64 hat der Benutzer bei allen Lade-Operationen Sorge zu tragen, daß genug Platz vorhanden ist. Sonst folgen undefinierte Resultate. (System-Absturz)

memtop (-- adr)
adr ist die erste Adresse oberhalb des verfügbaren RAM-Bereiches. Sie ist systemabhängig.

n" (--adr 8b)
Ist identisch der Sequenz
Ascii " parse
Der Quelltext bis inklusive dem nächsten " wird nicht ausgeführt, sondern als Zeichenkette verstanden. Sie beginnt bei der Adresse adr und ist 8b Bytes lang.

ramdisk (--)
Ein Vokabular, in sich Worte der Ramdisk befinden.

ramR/W (adr1 u adr2 flag--flag)
R/W wird bei Benutzung der Ramdisk auf RAMR/W gesetzt. Dadurch wird bei Blockzugriffen auf Drive 0 weiterhin auf ein Diskettenlaufwerk zugegriffen, Blockzugriffe auf Drive 1 und folgende werden jedoch auf die aktuelle Ramdisk umgeleitet. Zur Benutzung der Ramdisk vgl.: DRIVE >DRIVE DRV? COPY CONVEY

rd (--adr)
adr ist die Anfangsadresse der aktuellen Ramdisk. Wenn keine gültige Ramdisk eingerichtet ist, wird eine Fehlerbehandlung eingeleitet. Zum Format der Ramdisk vgl. die Shadows des Quelltext. Vgl. auch (RD

rdcheck (--)
Druckt Informationen über die Ramdisk aus und prüft die zentralen Zeiger.

rddel (--)
Löscht alle Blöcke der aktuellen Ramdisk.

rdnew (adr1 adr2--)
Richtet eine neue Ramdisk von adr1 bis maximal adr2-1 ein und setzt sie als die aktuelle. Eine Fehlerbedingung liegt vor, wenn der Speicherbereich von adr1 bis adr2 bereits anders belegt ist.

rduse (adr--)
Erklärt die Ramdisk ab adr zur aktuellen. Eine Fehlerbedingung liegt vor, wenn adr nicht die Anfangsadresse einer Ramdisk ist.

restore" (--)
Wirkt wie ABORT" , mit der Ausnahme, daß vorher mit STORE gesicherte Bereiche zurückgespeichert werden.

saveramdisk (--)
Die aktuelle Ramdisk wird auf das aktuelle Gerät gesichert. Eine Fehlerbedingung liegt vor, wenn keine Ramdisk eingerichtet war.

store (adr --)
Wirkt wie **PUSH**, mit der Ausnahme, daß bei einem anschließenden **RESTORE"** der gesicherte Bereich zurückgespeichert wird.

supertape (--)
Setzt den Kassettenrekorder im Supertape-Format als aktuelles Ausgabegerät. Supertape ist eine Schnell-Lade-Routine, die von der Zeitschrift "c't" für alle gängigen Mikro-Rechner angeboten wird. Wir danken dem Heise-Verlag für die freundliche Genehmigung, es in ultraFORTH83 integrieren und weiterverbreiten zu dürfen. Vgl.: **DEVICE COMMODORE FLOPPY**

tapeinit (--)
Initialisiert die Kassettenversion. Wenn autoload ungleich 0 ist, wird eine Ramdisk geladen. Wird im allgemeinen als **'RESTART** installiert.

Massenspeicher-Utilities

2disk1551 (--)
Sendet beim C16 einen Befehl über den Bus, der ein anwesendes Diskettenlaufwerk 1551 auf Unit 9 umstellt. Alle nicht gemeinten Laufwerke sind vorher auszuschalten.

bamallocate (--)
Kennzeichnet alle Sektoren einer Diskette als belegt.

copy2disk (--)
Kopiert eine Diskette von einem Laufwerk auf ein anderes. Die Directory wird mitkopiert, daher auch für Files zu verwenden.

copydisk (u1 u2 u3 --)
Kopiert die Blöcke u1 bis u2 einer Diskette ab Block u3 auf die andere Diskette.

formatdisk (--)
Formatiert eine Diskette für die Benutzung unter ultraFORTH83. Beim C16/C64 benutzt in der Form:

```
formatdisk [<name>,[<id>]]
```

Neue Disketten müssen mit ,[<id>] formatiert werden.
Beispiel:

```
formatdisk hallo,xx
```

savesystem (--)
Benutzt in der Form

```
savesystem <name>
```

Das ultraFORTH-System wird bis **HERE** auf einen externen Massenspeicher gesichert. Vorher wird es in den Zustand versetzt, in dem es beim nächsten Kaltstart sein soll. Insbesondere werden die aktuellen Parameter von Uservariablen in den User-Kaltstart-Bereich kopiert. Nicht gesichert werden Blockpuffer. S.a. "Erstellen eines eigenen Arbeitssystems".


Definition der Begriffe
FORTH-Gesellschaft eV
ultraFORTH83
Begriffe

Entscheidungskriterien

Bei Konflikten laesst sich das Standardteam von folgenden Kriterien in Reihenfolge ihrer Wichtigkeit leiten:

1. Korrekte Funktion - bekannte Einschraenkungen,Eindeutigkeit
2. Transportabilitaet - wiederholbare Ergebnisse, wenn Programme zwischen Standardsystemen portiert werden
3. Einfachheit
4. Klare, eindeutige Namen - die Benutzung beschreibender statt funktionaler Namen, zB [COMPILE] statt 'c, und ALLOT statt dp+!
5. Allgemeinheit
6. Ausfuehrungsgeschwindigkeit
7. Kompaktheit
8. Kompilationsgeschwindigkeit
9. Historische Kontinuitaet
10. Aussprechbarkeit
11. Verstaendlichkeit - es muss einfach gelehrt werden koennen

Adresse, Byte (address, byte)
Adresse, Kompilation (address, compilation)
Adresse, Natuerliche (address, native machine)
Adresse, Parameterfeld (address, parameter field) ""apf""
anzeigen (display)
Arithmetik, 2er-komplement (arithmetic, two's complement)
Block (block)
Blockpuffer (block buffer)
Byte (byte)
Kompilation (compilation)
Definition (Definition)
Dictionary (Woerterbuch)
Division, floored (division, floored)
Empfangen (receive)
Falsch (false)
Fehlerbedingung (error condition)
Flag (logischer Wert)
Floor, arithmetic
Glossar (glossary)
Interpreter, Adressen (interpreter, address)
Interpreter, Text (interpreter, text)
Kontrollstrukturen (structure, control)
laden (load)
Massenspeicher (mass storage)
Programm (program)
Quelltext (input stream)
Rekursion (recursion)
Screen (Bildschirm)
Suchreihenfolge (search order)
stack, data (Datenstapel)
stack, return (Ruecksprungstapel)
String, counted (abgezaehlte Zeichenkette)
String, Text (Zeichenkette)
Userarea (Benutzerbereich)
Uservariable (Benutzervariable)
Vokabular (vocabulary)
Vokabular, Kompilation (vocabulary, compilation)
Wahr (true)
Wort, Definierendes (defining word)
Wort, immediate (immediate word)
Wortdefinition (word definition)
Wortname (word name)
Zahl (number)
Zahlenausgabe, bildhaft (pictured numeric output)
Zahlenausgabe, freiformatiert (free field format)
Zahlentypen (number types)
Zahlenumwandlung (number conversion)
Zeichen (character)
Zustand (mode)

Definition der Begriffe

Es werden im allgemeinen die amerikanischen Begriffe beibehalten, es sei denn, der Begriff ist bereits gelauefig. Wird ein deutscher Begriff verwendet, so wird in Klammern der engl. Originalbegriff beigefuegt; wird der Originalbegriff beibehalten, so wird in Klammern eine moeglichst treffende Uebersetzung angegeben.

Adresse, Byte (address, byte)
: Eine 16bit Zahl ohne Vorzeichen, die den Ort eines 8bit Bytes im Bereich <0...65,535> angibt. Adressen werden wie Zahlen ohne Vorzeichen manipuliert.
Siehe: "Arithmetik, 2er-komplement"

Adresse, Kompilation (address, compilation)
: Der Zahlenwert, der zur Identifikation eines Forth Wortes kompiliert wird. Der Adressinterpreter benutzt diesen Wert, um den zu jedem Wort gehoerigen Maschinencode aufzufinden.

Adresse, Natuerliche (address, native machine)
: Die vorgegebene Adressdarstellung der Computerhardware.

Adresse, Parameterfeld (address, parameter field) ""apf""
: Die Adresse des ersten Bytes jedes Wortes, das fuer das Ablegen von Kompilationsadressen (bei :-definitionen) oder numerischen Daten bzw. Textstrings usw. benutzt wird.

anzeigen (display)
: Der Prozess, ein oder mehrere Zeichen zum aktuellen Ausgabegeraet zu senden. Diese Zeichen werden normalerweise auf einem Monitor angezeigt bzw. auf einem Drucker gedruckt.

Arithmetik, 2er-komplement (arithmetic, two's complement)
: Die Arithmetik arbeitet mit Zahlen in 2er-komplementdarstellung; diese Zahlen sind, je nach Operation, 16bit oder 32bit weit. Addition und Subtraktion von 2er-komplementzahlen ignorieren Ueberlaufsituationen. Dadurch ist es moeglich, dass die gleichen Operatoren benutzt werden koennen, gleichgueltig, ob man die Zahl mit Vorzeichen (<-32,768...32,767> bei 16bit) oder ohne Vorzeichen (<0...65,535> bei 16bit) benutzt.

Block (block)
: Die 1024byte Daten des Massenspeichers, auf die ueber Blocknummern im Bereich <0...Anzahl_existenter_Bloecke-1> zugegriffen wird. Die exakte Anzahl der Bytes, die je Zugriff auf den Massenspeicher uebertragen werden, und die Uebersetzung von Blocknummern in die zugehoerige Adresse des Laufwerks und des physikalischen Satzes, sind rechnerabhaengig.
Siehe: "Blockpuffer" und "Massenspeicher"

Blockpuffer (block buffer)
Ein 1024byte langer Hauptspeicherbereich, in dem ein Block voruebergehend benutzbar ist. Ein Block ist in hoechstens einem Blockpuffer enthalten.

Byte (byte)
Eine Einheit von 8bit. Bei Speichern ist es die Speicherkapazitaet von 8bits.

Kompilation (compilation)
Der Prozess, den Quelltext in eine interne Form umzuwandeln, die spaeter ausgefuehrt werden kann. Wenn sich das System im Kompilationszustand befindet, werden die Kompilationsadressen von Worten im Dictionary abgelegt, so dass sie spaeter vom Adresseninterpreter ausgefuehrt werden koennen. Zahlen werden so kompiliert, dass sie bei Ausfuehrung auf den Stack gelegt werden. Zahlen werden aus dem Quelltext ohne oder mit negativem Vorzeichen akzeptiert und gemaess dem Wert von BASE umgewandelt.
Siehe: "Zahl", "Zahlenumwandlung", "Interpreter, Text" und "Zustand"

Definition (Definition)
Siehe: "Wortdefinition"

Dictionary (Woerterbuch)
Eine Struktur von Wortdefinitionen, die im Hauptspeicher des Rechners angelegt ist. Sie ist erweiterbar und waechst in Richtung hoeherer Speicheradressen. Eintraege sind in Vokabularen organisiert, so dass die Benutzung von Synonymen moeglich ist, d.h. gleiche Namen koennen, in verschiedenen Vokabularen enthalten, vollkommen verschiedene Funktionen ausloesen.
Siehe: "Suchreihenfolge"

Division, floored (division, floored)
Ganzzahlige Division, bei der der Rest das gleiche Vorzeichen hat wie der Divisor oder gleich Null ist; der Quotient wird gegen die naechstkleinere ganze Zahl gerundet.
Bemerkung: Ausgenommen von Fehlern durch Ueberlauf gilt: N1 N2 SWAP OVER /MOD ROT * + ist identisch mit N1.
Siehe: "floor, arithmetisch"

Beispiele:	Dividend	Divisor	Rest	Quotient
	10	7	3	1
	-10	7	4	-2
	10	-7	-4	-2
	-10	-7	-3	1

Empfangen (receive)
Der Prozess, der darin besteht, ein Zeichen von der aktuellen Eingabeeinheit zu empfangen. Die Anwahl einer Einheit ist rechnerabhaengig.

Falsch (false)
Die Zahl Null repraesentiert den "Falschzustand" eines Flags.

Fehlerbedingung (error condition)
Eine Ausnahmesituation, in der ein Systemverhalten erfolgt, das nicht mit der erwarteten Funktion uebereinstimmt. Im der Beschreibung der einzelnen Worte sind die moeglichen Fehlerbedingungen und das dazugehoerige Systemverhalten beschrieben.

Flag (logischer Wert)
Eine Zahl, die eine von zwei moeglichen Werten hat, falsch oder wahr.
Siehe: "Falsch" "Wahr"

Floor, arithmetic
Z sei eine reelle Zahl. Dann ist der Floor von Z die groesste ganze Zahl, die kleiner oder gleich Z ist.
Der Floor von +0,6 ist 0
Der Floor von -0,4 ist -1

Glossar (glossary)
Eine umgangssprachliche Beschreibung, die die zu einer Wortdefinition gehoerende Aktion des Computers beschreibt - die Beschreibung der Semantik des Wortes.

Interpreter, Adressen (interpreter, address)
Die Maschinencodeinstruktionen, die die kompilierten Wortdefinitionen ausfuehren, die aus Kompilationsadressen bestehen.

Interpreter, Text (interpreter, text)
Eine Wortdefinition, die immer wieder einen Wortnamen aus dem Quelltext holt, die zugehoerige Kompilationsadresse bestimmt und diese durch den Adressinterpreter ausfuehren laesst. Quelltext, der als Zahl interpretiert wird, hinterlaesst den entsprechenden Wert auf dem Stack.
Siehe: "Zahlenumwandlung"

Kontrollstrukturen (structure, control)
Eine Gruppe von Worten, die, wenn sie ausgefuehrt werden, den Programmfluss veraendern.
Beispiele von Kontrollstrukturen sind:
DO ... LOOP
BEGIN ... WHILE ... REPEAT
IF ... ELSE ... THEN

laden (load)
Das Umschalten des Quelltextes zum Massenspeicher. Dies ist die uebliche Methode, dem Dictionary neue Definitionen hinzuzufuegen.

Massenspeicher (mass storage)
Speicher, der ausserhalb des durch FORTH adressierbaren Bereiches liegen kann. Auf den Massenspeicher wird in Form von 1024byte grossen Bloecken zugegriffen. Auf einen Block kann innerhalb des Forth-Adressbereichs in einem Blockpuffer zugegriffen werden. Wenn ein Block als veraendert (UPDATE) gekennzeichnet ist, wird er

letztendlich wieder auf den Massenspeicher zurueckgeschrieben.

Programm (program)
Eine vollstaendige Ablaufbeschreibung in FORTH-Quelltext, um eine bestimmte Funktion zu realisieren.

Quelltext (input stream)
Eine Folge von Zeichen, die dem System zur Bearbeitung durch den Textinterpreter zugefuehrt wird. Der Quelltext kommt ueblicherweise von der aktuellen Eingabeeinheit (ueber den Texteingabepuffer) oder dem Massenspeicher (ueber einen Blockpuffer). BLK, >IN, TIB und #TIB charakterisieren den Quelltext. Worte, die BLK, >IN, TIB oder #TIB benutzen und/oder veraendern, sind dafuer verantwortlich, die Kontrolle des Quelltextes aufrechtzuerhalten oder wiederherzustellen.
Der Quelltext reicht von der Stelle, die durch den Relativzeiger >IN angegeben wird, bis zum Ende. Wenn BLK Null ist, so befindet sich der Quelltext an der Stelle, die durch TIB adressiert wird, und er ist #TIB Bytes lang. Wenn BLK ungleich Null ist, so ist der Quelltext der Inhalt des Blockpuffers, der durch BLK angegeben ist, und er ist 1024byte lang.

Rekursion (recursion)
Der Prozess der direkten oder indirekten Selbstreferenz.

Screen (Bildschirm)
Ein Screen sind Textdaten, die zum Editieren aufbereitet sind. Nach Konvention besteht ein Screen aus 16 Zeilen zu je 64 Zeichen. Die Zeilen werden von 0 bis 15 durchnumeriert. Screens enthalten normalerweise Quelltext, koennen jedoch auch dazu benutzt werden, um Massenspeicherdaten zu betrachten. Das erste Byte eines Screens ist gleichzeitig das erste Byte eines Massenspeicherblocks; dies ist auch der Anfangspunkt fuer Quelltextinterpretation waehrend des Ladens eines Blocks.

Suchreihenfolge (search order)
Eine Spezifikation der Reihenfolge, in der ausgewaehlte Vokabulare im Dictionary durchsucht werden. Die Suchreihenfolge besteht aus einem auswechselbaren und einem festen Teil, wobei der auswechselbare Teil immer als erstes durchsucht wird. Die Ausfuehrung eines Vokabularnamens macht es zum ersten Vokabular in der Suchreihenfolge, wobei das Vokabular, das vorher als erstes durchsucht worden war, verdraengt wird. Auf dieses erste Vokabular folgt, soweit spezifiziert, der feste Teil der Suchreihenfolge, der danach durchsucht wird. Die Ausfuehrung von ALSO uebernimmt das Vokabular im auswechselbaren Teil in den festen Teil der Suchreihenfolge. Das Dictionary wird immer dann durchsucht, wenn ein Wort durch seinen Namen aufgefunden werden soll.

stack, data (Datenstapel)
Eine "Zuletzt-rein, Zuerst-raus" (last-in, first-out) Struktur, die aus einzelnen 16bit Daten besteht. Dieser Stack wird hauptsaechlich zum Ablegen von Zwischenergebnissen waehrend des Ausfuehrens von Wortdefinitionen benutzt. Daten auf dem Stack koennen Zahlen, Zeichen, Adressen, Boole'sche Werte usw. sein. Wenn der Begriff "Stapel" oder "Stack" ohne Zusatz benutzt wird, so ist immer der Datenstack gemeint.

stack, return (Ruecksprungstapel)
Eine "Zuletzt-rein, Zuerst-raus" Struktur, die hauptsaechlich Adressen von Wortdefinitionen enthaelt, deren Ausfuehrung durch den Adressinterpreter noch nicht beendet ist. Wenn eine Wortdefinition eine andere Wortdefinition aufruft, so wird die Ruecksprungadresse auf dem Returnstack abgelegt.
Der Returnstack kann zeitweise auch fuer die Ablage anderer Daten benutzt werden.

String, counted (abgezaehlte Zeichenkette)
Eine Hintereinanderfolge von 8bit Daten, die im Speicher durch ihre niedrigste Adresse charakterisiert wird. Das Byte an dieser Adresse enthaelt einen Zahlenwert im Bereich <0...255>, der die Anzahl der zu diesem String gehoerigen Bytes angibt, die unmittelbar auf das Countbyte folgen. Die Anzahl beinhaltet nicht das Countbyte selber. Counted Strings enthalten normalerweise ASCII-Zeichen.

String, Text (Zeichenkette)
Eine Hintereinanderfolge von 8bit Daten, die im Speicher durch ihre niedrigste Adresse und ihre Laenge in Bytes charakterisiert ist. Strings enthalten normalerweise ASCII-Zeichen. Wenn der Begriff "String" alleine oder in Verbindung mit anderen Begriffen benutzt wird, so sind Textstrings gemeint.

Userarea (Benutzerbereich)
Ein Gebiet im Speicher, das zum Ablegen der Uservariablen benutzt wird und fuer jeden einzelnen Prozess/Benutzer getrennt vorhanden ist.

Uservariable (Benutzervariable)
Eine Variable, deren Datenbereich sich in der Userarea befindet. Einige Systemvariablen werden in der Userarea gehalten, so dass die Worte, die diese benutzen, fuer mehrere Prozesse/Benutzer gleichzeitig verwendbar sind.

Vokabular (vocabulary)
Eine geordnete Liste von Wortdefinitionen. Vokabulare werden vorteilhaft benutzt, um Worte voneinander zu unterscheiden, die gleiche Namen haben (Synonyme). In einem Vokabular koennen mehrere Definitionen mit dem gleichen Namen existieren; diesen Vorgang nennt man redefinieren. Wird das Vokabular nach einem Namen durchsucht, so wird die juengste Redefinition gefunden.

Vokabular, Kompilation (vocabulary, compilation)
Das Vokabular, in das neue Wortdefinitionen eingetragen werden.

Wahr (true)
Ein Wert, der nicht Null ist, wird als "wahr" interpretiert. Wahrwerte, die von Standard-FORTH-Worten errechnet werden, sind 16bit Zahlen, bei denen alle 16 Stellen auf "1" gesetzt sind, so dass diese zum Maskieren benutzt werden koennen.

Wort, Definierendes (defining word)
Ein Wort, das bei Ausfuehrung einen neuen Dictionary-Eintrag im Kompilationsvokabular erzeugt. Der Name des neuen Wortes wird dem Quelltext entnommen. Wenn der Quelltext erschoepft ist, bevor der neue Name erzeugt wurde, so wird die Fehlermeldung "ungueltiger Name" ausgegeben.
Beispiele von definierenden Worten sind:
: CONSTANT CREATE

Wort, immediate (immediate word)
Ein Wort, das ausgefuehrt wird, wenn es waehrend der Kompilation oder Interpretation aufgefunden wird. Immediate Worte behandeln Sondersituationen waehrend der Kompilation.
Siehe z.B. IF LITERAL ." usw.

Wortdefinition (word definition)
Eine mit einem Namen versehene, ausfuehrbare FORTH-Prozedur, die ins Dictionary kompiliert wurde. Sie kann durch Maschinencode, als eine Folge von Kompilationsadressen oder durch sonstige kompilierte Worte spezifiziert sein. Wenn der Begriff "Wort" ohne Zusatz benutzt wird, so ist im allgemeinen eine Wortdefinition gemeint.

Wortname (word name)
Der Name einer Wortdefinition. Wortnamen sind maximal 31 Zeichen lang und enthalten kein Leerzeichen. Haben zwei Definitionen verschiedene Namen innerhalb desselben Vokabulars, so sind sie eindeutig auffindbar, wenn das Vokabular durchsucht wird.
Siehe: "Vokabular"

Zahl (number)
Wenn Werte innerhalb eines groesseren Feldes existieren, so sind die hoeherwertigen Bits auf Null gesetzt. 16bit Zahlen sind im Speicher so abgelegt, dass sie in zwei benachbarten Byteadressen enthalten sind. Die Bytereihenfolge ist rechnerabhaengig. Doppeltgenaue Zahlen (32bit) werden auf dem Stack so abgelegt, dass die hoeherwertigen 16bit (mit dem Vorzeichenbit) oben liegen. Die Adresse der niederwertigen 16bit ist um zwei groesser als die Adresse der hoeherwertigen 16bit, wenn die Zahl im Speicher abgelegt ist.
Siehe: "Arithmetik, 2er-komplement" und "Zahlentypen"

Zahlenausgabe, bildhaft (pictured numeric output)
Durch die Benutzung elementarer Worte fuer die Zahlenausgabe (z.B. <# # #s #>) werden Zahlenwerte in Textstrings umgewandelt. Diese Definitionen werden in einer Folge benutzt, die ein symbolisches Bild des gewuenschten Ausgabeformates darstellen. Die Umwandlung schreitet von der niedrigstwertigen zur hoechstwertigen Ziffer fort und die umgewandelten Zeichen werden von hoeheren gegen niedrigere Speicheradressen abgelegt.

Zahlenausgabe, freiformatiert (free field format)
Zahlen werden in Abhaengigkeit von BASE umgewandelt und ohne fuehrende Nullen, aber mit einem folgenden Leerzeichen, angezeigt. Die Anzahl von Stellen, die angezeigt werden, ist die Minimalanzahl von Stellen - mindestens eine - die notwendig sind, um die Zahl eindeutig darzustellen.
Siehe: "Zahlenumwandlung"

Zahlentypen (number types)
Alle Zahlentypen bestehen aus einer spezifischen Anzahl von Bits. Zahlen mit oder ohne Vorzeichen bestehen aus bewerteten Bits. Bewertete Bits innerhalb einer Zahl haben den Zahlenwert einer Zweierpotenz, wobei das am weitesten rechts stehende Bit (das niedrigstwertige) einen Wert von zwei hoch null hat. Diese Bewertung setzt sich bis zum am weitesten links stehenden Bit fort, wobei sich die Bewertung fuer jedes Bit um eine Zweierpotenz erhoeht. Fuer eine Zahl ohne Vorzeichen ist das am weitesten links stehende Bit in diese Bewertung eingeschlossen, so dass fuer eine solche 16bit Zahl das ganz linke Bit den Wert 32.768 hat. Fuer Zahlen mit Vorzeichen wird die Bewertung des ganz linken Bits negiert, so dass es bei einer 16bit Zahl den Wert -32.768 hat. Diese Art der Wertung fuer Zahlen mit Vorzeichen wird 2er-komplementdarstellung genannt.
Nicht spezifizierte, bewertete Zahlen sind mit oder ohne Vorzeichen; der Programmkontext bestimmt, wie die Zahl zu interpretieren ist.

Zahlenumwandlung (number conversion)
Zahlen werden intern als Binaerzahlen gefuehrt und extern durch graphische Zeichen des ASCII Zeichensatzes dargestellt. Die Umwandlung zwischen der internen und externen Form wird unter Beachtung des Wertes von BASE durchgefuehrt, um die Ziffern einer Zahl zu bestimmen. Eine Ziffer liegt im Bereich von Null bis BASE-1. Die Ziffer mit dem Wert Null wird durch das ASCII-zeichen "0" (Position 3/0, dezimalwert 48) dargestellt. Diese Zifferndarstellung geht den ASCII-code weiter aufwaerts bis zum Zeichen "9", das dem dezimalen Wert neun entspricht. Werte, die jenseits von neun liegen, werden durch die ASCII-zeichen beginnend mit "A", entsprechend dem Wert zehn usw. bis zum ASCII-zeichen "~", entsprechend einundsiebzig, dargestellt.Bei einer negativen Zahl wird das ASCII-zeichen "-" den Ziffern vorangestellt. Bei der Zahleneingabe kann der aktuelle Wert von BASE fuer die gerade umzuwandelnde Zahl dadurch umgangen werden, dass den Ziffern ein "Zahlenbasisprefix" vorangestellt wird. Dabei wird durch das Zeichen "%" die Basis voruebergehend

auf den Wert zwei gesetzt, durch "&" auf den Wert zehn und durch "$" oder "h" auf den Wert sechzehn. Bei negativen Zahlen folgt das Zahlenbasisprefix dem Minuszeichen. Enthaelt die Zahl ein Komma oder einen Punkt, so wird sie als 32bit Zahl umgewandelt.

Zeichen (character)
Eine 7bit Zahl, deren Bedeutung durch den ASCII-Standard festgelegt ist. Wenn es in einem groesseren Feld gespeichert ist, so sind die hoeherwertigen Bits auf Null gesetzt.

Zustand (mode)
Der Textinterpreter kann sich in zwei moeglichen Zustaenden befinden: dem interpretierenden oder dem kompilierenden Zustand. Die Variable STATUS wird vom System entsprechend gesetzt, und zwar enthaelt sie bei der Interpretation eine False-flag, bei der Kompilation eine True-flag.
Siehe: "Interpreter, Text" und "Kompilation"